UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF WASHINGTON

DATANET LLC, Plaintiff, v. MICROSOFT CORPORATION, Defendant.	NO. 2:22-cv-01545 **COMPLAINT FOR PATENT INFRINGEMENT** **JURY DEMAND**

Plaintiff Datanet LLC ("Datanet" or "Plaintiff"), by and through its attorneys, for its Complaint against defendant Microsoft Corporation ("Defendant") alleges as follows:

NATURE OF THE ACTION

1. This is a civil action for patent infringement under the patent laws of the United States, 35 U.S.C. § 1 et seq.

2. This action arises from Defendant's unlicensed use of Datanet's patented technology in Defendant's file hosting/backup service called Microsoft OneDrive ("OneDrive"). The marketing, testing, importation, use and sale of such product(s) within the United States infringes upon the intellectual property rights of Datanet. This illegal practice will continue unless, and until, the Court puts an end to it.

3. By this action Plaintiff seeks money damages, exemplary damages and attorneys' fees arising from Defendant's patent infringement under the Patent Act, 35 U.S.C. §§ 271, 283–285.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

PARTIES

4. Plaintiff is a limited liability company formed under the laws of Nevada with a principal place of business located in Colorado.

5. Defendant is a corporation organized under the laws of Washington.

6. Defendant maintains its headquarters at 1 Microsoft Way, Redmond, Washington.

JURISDICTION AND VENUE

7. This is an action for infringement against Defendants brought under the Patent Act, 35 U.S.C. § 271 based upon Defendants' unauthorized commercial manufacture, testing, use, importation, offer for sale and sale of its file hosting/backup service called OneDrive which infringes upon United States ("U.S.") Patent Numbers 8,473,478 ("'478 Patent), and 9,218,348 ("'348 Patent"), and 10,585,850 ("'850 Patent).

8. This Court has subject matter jurisdiction over the matters asserted herein under 28 U.S.C. §§ 1331 and 1338(a).

9. This court has personal jurisdiction over Defendant because Defendant is a Washington Corporation that maintains its headquarters in Redmond, Washington, has a significant presence in Washington, conducts its business throughout the United States, including within the state of Washington, and has committed in this District the acts of patent infringement, which have given rise to this action.

10. Venue is proper in this District under 28 U.S.C. §§ 1391 and 1400(b) as Defendant is a Washington Corporation that maintains its headquarters in Redmond, Washington, has a significant presence in this District, and Defendant has advertised and derived revenue from sales of products to citizens within this District and has engaged in systematic and continuous business contact within this State. Defendant has had and continues to have significant contact with the state of Washington through its U.S. based websites, through U.S. based sales, and distribution of its OneDrive product and services throughout the U.S., and have purposefully availed themselves of Washington's laws.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

11. Upon information and belief, a number of Defendant's current and former employees with information relevant to this case reside in the Redmond, Washington area.

BACKGROUND

A. Plaintiff's Patented Technology

12. The use and importance of data backup and storage, and cloud storage in particular, have grown tremendously since the year 2000.

13. In 1999, a company called IPCI, Inc. ("IPCI") set out to develop an automated, real-time zero-touch data safety, backup, and recovery software product. It called the project AccuSafe.

14. Unfortunately, the AccuSafe product was never marketed or sold, however, though the development process, IPCI secured a portfolio of patents (the "Patent Portfolio") around the unique and novel technology.

15. On September 20, 2018, IPCI assigned its rights to the Patent Portfolio to Datanet, LLC ("Datanet").

16. At a high level, the Patent Portfolio protects user files and describes systems and techniques for archiving and restoring files.

17. The Patent Portfolio consists of five issued U.S. Patents, U.S. Patent Nos. 8,473,478, 9,218,348, 10,229,123, 10,331,619, and 10,585,850.

18. U.S. Patent No. 8,473,478 (the "'478 Patent") was filed on September 21, 2001 and claims priority to U.S. Provisional Patent Application No. 60/234,221, filed on September 21, 2000.

19. The '478 Patent was granted an extended term through October 1, 2026.

20. U.S. Patent No. 9,218,348 (the "'348 Patent") was filed on June 24, 2013 as a continuation of the '478 Patent.

21. U.S. Patent No. 10,585,850 (the "'850 Patent") was filed on December 15, 2015 as a continuation of the '348 Patent.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

22. The ’850 Patent was granted an extended term though June 25, 2024.

23. U.S. Patent No. 10,229,123 (the “’123 Patent”) was filed on October 20, 2017 as a continuation of U.S. Patent No. 10,585,850.

24. U.S. Patent No. 10,331,619 (the “’619 Patent”) was filed as a continuation of U.S. Patent No. 10,585,850.

25. Before the development of the AccuSafe project, file backup procedures (manual backup, schedule-based backup, and mirroring) were lacking in important ways, including: (1) they were complex or confusing for users, (2) they did not record all the changes made and often could not be reversed, and (3) they required the backup medium to be available at the time of backup.

26. The Datanet Patent Portfolio provides significant advantages to the previous file backup systems by providing, *inter alia*: (1) the ability to archive files, or portions of the files, in close proximity of time to opening, closing, or saving the files, allowing for timely and full backup for later recovery, (2) the ability to store files (or portions of files) for remote or cloud backup when the storage location becomes available, and (3) the ability to preview prior versions of a file before restoring a prior version, so that only desired backup versions of the files are restored.

27. The claims of the Datanet portfolio are directed to improvements in computer functionality when applied to data storage, and address the failings that are unique to data storage networked environments. The Specifications of the patents in suit disclose that the inventions relates to computer-specific improvements, viz. “file preservation, file capture, file management, and file integrity techniques for computing devices,” and addresses computer-specific problems that arise given the ever-changing networked-data environment, through “captur[ing] files just before and/or just after the files are changed,” “hav[ing] an imperceptible impact on system performance from the user’s point of view,” and “captur[ing] and stor[ing] files even when there

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

is no connection to the desired storage location … [and/or] when the desired storage location is unavailable."

28. The '478 Patent solution improves computer functionality over prior data storage approaches because it optimizes the use of various storage locations to capture changes to files in real time (or near real time), along with database(s) to track the movement of the files between the storage locations, so that previous versions of file(s) can be efficiently retrieved and restored, without overburdening the network resources in the process.

29. The claims of the '478 patent recite specific interactions between hardware and software computer components to accomplish the data backup and storage, and through these interactions, the computer functionality is improved. The inventions claimed in the '478 patent save user's time, computer resources, and network bandwidth over prior approaches.

30. The '348 Patent solution improves computer functionality over prior data storage approaches because it optimizes the use of various storage locations to capture changes to files in real time (or near real time), along with database(s) to track the movement of the files between the storage locations, so that previous versions of file(s) can be efficiently retrieved and restored, without overburdening the network resources in the process.

31. The claims of the '348 patent recite specific interactions between hardware and software computer components to accomplish the data backup and storage, and through these interactions, the computer functionality is improved. The inventions claimed in the '348 patent save user's time, computer resources, and network bandwidth over prior approaches.

32. The '850 Patent solution improves computer functionality over prior data storage approaches because it optimizes the use of various storage locations to capture changes to files in real time (or near real time), along with database(s) to track the movement of the files between the storage locations, so that previous versions of file(s) can be quickly previewed before being efficiently retrieved and restored, all without overburdening the network resources in the process.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

33. The claims of the '850 patent recite specific interactions between hardware and software computer components to accomplish the data backup, presentation of previews, and the storage of such, and through these interactions, the computer functionality is improved. The inventions claimed in the '850 patent save user's time, computer resources, and network bandwidth by avoiding the restoration of an unwanted previous version. The technique of previewing multiple previous versions prior to restoring from network storage was not conventional and was not well-understood at the time of the invention.

B. Defendant's Infringement of the Asserted Patents

34. Defendant was founded in 1975.

35. Upon information and belief, Defendant launched its flagship services called OneDrive and OneDrive for Business in August 2007 (collectively referred herein as "OneDrive").

36. OneDrive provides users with the ability to share, synchronize and backup their files.

37. Upon information and belief, OneDrive is bundled with Microsoft Windows and is available for use on a variety of platforms including Android, iOS, macOS, Windows Phone, Xbox 360, Xbox One, Xbox Series X, and Xbox Series S.

38. In addition, upon information and belief, Microsoft Office directly integrates with OneDrive.

39. Upon information and belief, OneDrive was called SkyDrive before the name was changed in 2013.

40. Upon information and belief, OneDrive for Business was called SkyDrive Pro before the name was changed in 2013.

41. Upon information and belief, since 2015, Defendant has provided a set amount of free storage to each OneDrive user, which has typically ranged somewhere between five (5) and fifteen (15) gigabytes.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

42. Upon information and belief, each user of OneDrive may purchase up to one (1) terabyte of storage space on Defendant's server(s).

43. OneDrive provides the ability for a user to save previous versions of a file and then gives them the ability to restore a current version of a file to a previous version of the file by selecting one of the saved previous versions of the file.

44. Upon information and belief, prior to July 2017, OneDrive only saved previous versions of files for Microsoft Office files.

45. In July 2017, OneDrive expanded its version history support to all file types.

46. Older versions of files are stored for up to thirty (30) days with OneDrive.

47. OneDrive uses the inventions claimed in the '348, '478 and '850 Patents to provide file backup capabilities to more than 42.7 million registered users of Microsoft Office.

48. OneDrive provides users with the ability to archive files in close proximity of time to opening, updating, closing, or saving them.

49. Upon information and belief, OneDrive maintains servers for storage of its user data located in Redmond, Washington.

50. OneDrive provides users with the ability to store files for remote backup when storage media is available.

51. OneDrive protects user files from unwanted edits, deletions, hackers, and viruses by restoring or recovering files in each OneDrive account for up to thirty (30) days.

52. OneDrive provides users with the ability to preview prior versions of a file before restoring it.

53. OneDrive provides users with a viewer history.

FIRST CLAIM FOR RELIEF

(Infringement of the '478 Patent – 35 U.S.C. §271)

54. Plaintiff hereby incorporates by reference and realleges the foregoing paragraphs as if fully set forth herein.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

55. Plaintiff is the current exclusive owner and assignee of all right, title, and interest in and to the ’478 Patent titled “Automatic Real-Time File Management Method and Apparatus”.

56. The ‘478 Patent was duly and legally issued by the United States Patent and Trademark Office on June 25, 2013.

57. Plaintiff owns all rights to the ’478 Patent including the right to bring this suit for damages.

58. A true and correct copy of the ’478 patent is attached hereto as Exhibit A.

59. The ’478 Patent is valid and enforceable.

60. Defendant has directly and indirectly infringed the ’478 patent by making, using, testing, selling, offering for sale, and/or importing into the United States, without authority, products, methods performed by and/or attributable to equipment, and or services that practice one or more claims of the ’478 Patent, including but not limited to its OneDrive file hosting product and services.

61. As a non-limiting example, Defendant has infringed Claims 1, 2, 3, 5, 6, 8, 9, 10 and 11 of the ’478 Patent. Defendant, without authorization from Plaintiff, has marketed, used, tested, offered for sale, sold, and/or imported into the U.S., including within this District, its OneDrive file hosting service that infringes at least Claims 1, 2, 3, 5, 6, 8, 9, 10 and 11 of the ‘478 Patent.

62. As a non-limiting example, set forth below (with claim language in italics) is a description of infringement of exemplary Claims 1 and 6 of the ’478 Patent in connection with OneDrive. This description is based on publicly available information. Plaintiff reserves the right to modify this description, including, for example, on the basis of information about OneDrive that it obtains during discovery.

Claim 1.

Claim 1. In a computing device, a method for archiving files comprising:	OneDrive automatically backs up local files (i.e., operating files and/or entire folders) to the one drive

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

cloud/server. When a file is added to the local computer OneDrive folder, a copy of the file (i.e. Archive file) is created in a temporary storage staging buffer (change to send queue) to facilitate sending to the cloud/server. The file in the OneDrive server is an archive file of the operating file, synchronized from the local file. If any changes are made to the local file, the changes get synchronized to the OneDrive archive file. Https://support.office.com/en-us/article/sync-files-with-onedrive-in-windows-615391c4-2bd3-4aae-a42a-858262e42a49?ui=en-us&rs=en-us&ad=us. The OneDrive service monitors the OneDrive folder on your computer and/or your OneDrive mobile app. If there's a change – a new file or folder, or an edit to an existing file of folder – OneDrive will automatically sync those changes. No manual uploading or downloading is required. Https://support.office.com/en-us/article/sync-files-with-onedrive-in-windows-615391c4-2bd3-4aae-a42a-858262e42a49?ui=en-us&rs=en-us&ad=us. The method steps for archiving files using OneDrive are performed on a computing device such as a computer or handheld phone.

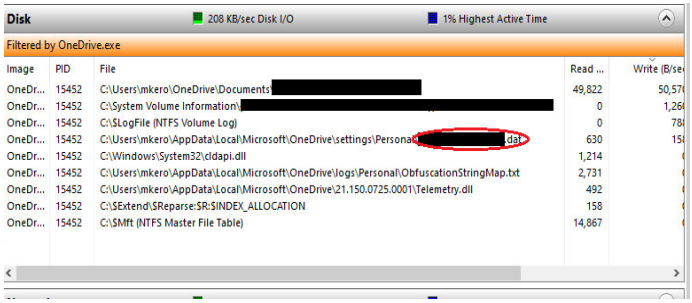
BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

Detecting an instruction by an operating system to perform an operation on an operating file;	OneDrive monitors changes made in files, and upon detecting a change in a file, OneDrive provides an instruction to save the modified file as an archived file. OneDrive detects the instruction to change/save the file. The windows notification service monitors the users file and provides an instruction upon detecting a change in a file, and OneDrive detects the instruction from the windows notification system to start the back-up process. http://go.microsoft.com/fwlink/p/?linkid=829044
Creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;	After detection of, and in response to the instruction to begin the back-up process, which occurs as soon as or shortly after a change in the file is made, a portion or all of the operating file is processed to create an archive file which is saved in a "change to send" queue. OneDrive uses windows notification system to detect and archive changes to files in real time. The windows notification system informs the OneDrive sync application to activate whenever a change to a file actually happens. Depending on the changes being made to a file, OneDrive may break the file into different sized pieces to create the archive file that is temporarily stored in the change to send queue (first temporary storage location). http://go.microsoft.com/fwlink/p/?linkid=829044.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

Searching the first temporary storage location for the archive file responsive to the occurrence of a first event; and	Once the archive file is moved into the change to send queue (first event), the archive files are identified so that they can be searched to determine their size/types before being sent to the OneDrive cloud server. This identification process can be the first event, or the first event can be a message from the operating process that created the archive file to indicate it is done, or a message from a timer to search the change to send queue. In response to the first event, the change to send queue (first temporary storage location) is searched in order to determine the archive file size/type prior to being sent to the OneDrive cloud server. Http://go.microsoft.com/fwlink/p/?linkid=829044.
Moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location;	The files (i.e., archived files) are transmitted (moved) from a user's computer (i.e., local device) to the OneDrive cloud server (the second, permanent storage location). When the identified archived file has a larger size, then it is divided into chunks for uploading. After dividing the file into smaller chunks (if needed), one by one, these chunks are transmitted from the change to send queue (first temporary storage) to the OneDrive cloud server (permanent storage) in response to a different second event that indicates the OneDrive server is ready to receive data. File transmission only occurs in response to a

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

	message from the OneDrive cloud server that it can receive data (second event). http://go.microsoft.com/fwlink/p/?linkid=829044. https://support.office.com/en-us/article/change-the-onedrive-sync-app-upload-or-download-rate-71cc69da-2371-4981-8cc8-b4558bdda56e?ui=en-us&rs=en-us&ad=us.
After storing the archive file in the first temporary storage location, updating a database to indicate that the archive file is located in the first temporary storage location;	After storing the archive file in the change to send queue, OneDrive updates a OneDrive database, upon information and belief "<cid>.dat" database, "metadata" database and/or "auto_upload.db" database to indicate the archive file is in the first temporary storage location. Http://go.microsoft.com/fwlink/p/?linkid=829044 (see also [private app storage path]/database/metadata; [private app storage path]/database/auto_upload.db) 
Determining a final destination for the archive file;	The archive file is determined and stored in one or more encrypted file chunks in the azure storage as the final destination, with metadata pointers to the azure storage on the OneDrive cloud server.

BYRNES ♦ KELLER ♦ CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

	http://go.microsoft.com/fwlink/p/?linkid=829044.
Moving the archive file from the first temporary storage location to an intermediate storage location;	OneDrive moves the archive file from the change to send queue to an output buffer, or a local cache, an external cache, or an intermediary server (intermediate storage location); before it moves the archive file into a final destination such as the azure storage on the OneDrive cloud server.
Updating the database to indicate that the archive file is located in the intermediate storage location; and	The OneDrive database is updated with "loadingprogress" data which tracks whether data transfer to the cloud has taken place for any given archive file, the progress of such transfer, and the progress of the output buffer, the local cache, the external cache, or the intermediary server.
After moving the archive file to the second storage location, updating the database to indicate that the archive file is located in the second storage location.	When an archive file is being moved to the azure storage, OneDrive shows a blue icon to indicate the archive file is being transferred to azure storage (i.e., from the output buffer, the local cache, the external cache, or the intermediary server). When the archive file has been moved into the azure storage (i.e., has been transferred to the final destination), the OneDrive database is updated and a green icon is shown at the client side instead of the blue icon.
Claim 6.	
An article of manufacture comprising a non-transitory computer useable medium having computer readable	Defendant hosts a computer server storing a computer executable application (called OneDrive) that has computer readable code that performs the

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

code for performing the method of claim 1.	steps of claim 1, as detailed above. (https://www.microsoft.com/en-us/microsoft-365/onedrive/download).

63. As described above, Defendant's products or services infringe the '478 Patent.

64. Defendant's OneDrive service includes a program which can be installed on a computer.

65. Defendant's OneDrive program detects instructions to perform operation(s) on files.

66. When an instruction is detected, Defendant's OneDrive program creates an archived version of the file being stored or modified and stores the archives file in a temporary storage location.

67. Defendant's OneDrive service records in a database the location of the archived file in the temporary storage location.

68. Later, the archived file is moved to a second storage location.

69. Defendant's OneDrive service records in the database the location of the archived file in the second storage location.

70. Defendant has actual knowledge of Plaintiff's rights in the '478 Patent and details of OneDrive's infringement of the '478 Patent based on at least the filing and service of this Complaint.

71. By the foregoing acts, including its testing and use of OneDrive, Defendant has directly infringed, literally and/or under the doctrine of equivalents, the '478 Patent in violation of 35 U.S.C. § 271.

72. Defendant has knowledge of the '478 Patent and indirectly infringes the '478 Patent by active inducement under 35 U.S.C. § 271(b) and/or § 271(f). Defendant has induced, caused, urged, encouraged, aided and abetted its direct and indirect customers to make, use, test, sell, offer for sale and/or import accused products and/or services. Defendant has done so by acts

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

including but not limited to selling accused products and/or services to its customers; marketing accused products and/or services; and providing instructions, technical support, and other support and encouragement (available at https://support.microsoft.com/en-us/office/back-up-your-documents-pictures-and-desktop-folders-with-onedrive-d61a7930-a6fb-4b95-b28a-6552e77c3057, or https://support.microsoft.com/en-US/search/results?query=restore+file+from+backup+OneDrive&isEnrichedQuery=true, for instance) for the use of accused products and/or services. Such conduct by Defendant was intended to and actually resulted in direct infringement, including the making, testing, using, selling, offering for sale, and/or importation of accused products and/or services in the United States.

73. Defendant has knowledge of the '478 Patent and indirectly infringes by contributing to the infringement of, and continuing to contribute to the infringement of, one or more claims of the '478 Patent under 35 U.S.C. § 271(c) and/or 271(f) by selling, offering for sale, and/or importing into the United States, the accused products and/or services. Defendant knows that the accused products and/or services include hardware components and software instructions that work in concert to perform specific, intended functions. Such specific, intended functions, carried out by these hardware and software combinations, are a material part of the inventions of the '478 Patent and are not staple articles of commerce suitable for substantial non-infringing use.

74. Upon information and belief, Defendant's infringement has, and continues to be, knowing, intentional and willful.

75. Defendant's acts of infringement of the '478 Patent has caused, and will continue to cause, Plaintiff damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284.

76. Upon information and belief, Defendant has gained profits by virtue of its infringement of the '478 Patent.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

77. The circumstances surrounding Defendant's infringement are exceptional and, therefore, Plaintiff is entitled to an award of attorney's fees pursuant to 35 U.S.C. § 285.

SECOND CLAIM FOR RELIEF

(Infringement of the '348 Patent – 35 U.S.C. §271)

78. Plaintiff hereby incorporates by reference and realleges the foregoing paragraphs as if fully set forth herein.

79. Plaintiff is the current exclusive owner and assignee of all right, title, and interest in and to the '348 Patent titled "Automatic Real-Time File Management Method and Apparatus".

80. The '348 Patent was duly and legally issued by the United States Patent and Trademark Office on December 22, 2015.

81. Plaintiff owns all rights to the '348 Patent including the right to bring this suit for damages.

82. A true and correct copy of the '348 Patent is attached hereto as Exhibit B.

83. Before its expiration, the '348 Patent was valid and enforceable.

84. Defendant has directly and indirectly infringed the '348 patent by making, using, testing, selling, offering for sale, and/or importing into the United States, without authority, products, methods performed by and/or attributable to equipment, and or services that practice one or more claims of the '348 Patent, including but not limited to its OneDrive file hosting product and services.

85. As a non-limiting example, Defendant has infringed Claims 1, 3, 4, 5, 6, 8, 10 – 20, and 23 – 31 of the '348 Patent. Defendant, without authorization from Plaintiff, has marketed, tested, used, offered for sale, sold, and/or imported into the U.S., including within this District, its OneDrive file hosting service that infringes at least Claims 1, 3, 4, 5, 6, 8, 10 – 20, and 23 – 31 of the '348 Patent.

86. As a non-limiting example, set forth below (with claim language in italics) is a description of infringement of exemplary Claims 15 and 29 of the '348 Patent in connection with

BYRNES ♦ KELLER ♦ CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

OneDrive. This description is based on publicly available information. Plaintiff reserves the right to modify this description, including, for example, on the basis of information about OneDrive that it obtains during discovery.

Claim 15

A method for archiving files, comprising of steps (a) to (d) following:	OneDrive automatically backs up local files (i.e., operating files and/or entire folders) to the one drive cloud/server. When a file is added to the local computer OneDrive folder, a copy of the file (i.e. Archive file) is created in a temporary storage staging buffer (change to send queue) to facilitate sending to the cloud/server. The file in the OneDrive server is an archive file of the operating file, synchronized from the local file. If any changes are made to the local file, the changes get synchronized to the OneDrive archive file. Https://support.office.com/en-us/article/sync-files-with-onedrive-in-windows-615391c4-2bd3-4aae-a42a-858262e42a49?ui=en-us&rs=en-us&ad=us. The OneDrive service monitors the OneDrive folder on your computer and/or your OneDrive mobile app. If there's a change – a new file or folder, or an edit to an existing file of folder – OneDrive will automatically sync those changes. No manual uploading or downloading is required. Https://support.office.com/en-us/article/sync-files-

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

	with-onedrive-in-windows-615391c4-2bd3-4aae-a42a-858262e42a49?ui=en-us&rs=en-us&ad=us. The method steps for archiving files using OneDrive are performed on a computing device such as a computer or handheld phone.
(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file;	OneDrive monitors changes made in files, and upon detecting a change in a file, OneDrive provides an instruction to save the modified file as an archived file. OneDrive detects the instruction to save the file. The windows notification service monitors the users file and provides an instruction upon detecting a change in a file, and OneDrive detects the instruction from the windows notification system to start the back-up process. http://go.microsoft.com/fwlink/p/?linkid=829044
(b) the step of creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;	After detection of, and in response to the instruction to begin the back-up process, which occurs as soon as or shortly after a change in the file is made, a portion or all of the operating file is processed to create an archive file which is saved to a "change to send" queue. OneDrive uses windows notification system to detect and archive changes to files in real time. The windows notification system informs the OneDrive sync application to activate whenever a change to a file actually happens. Depending on the changes being

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

	made to a file, OneDrive may break the file into different sized pieces to create the archive file that is temporarily stored in the change to send queue (first temporary storage location). http://go.microsoft.com/fwlink/p/?linkid=829044.
(c) the step of identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event; and	Once the archive file is moved into the change to send queue (first event), the archive files are identified so that they can be searched to determine their size/types before being sent to the OneDrive cloud server. This identification process can be the first event, or the first event can be a message from the operating process that created the archive file to indicate it is done, or a message from a timer to search the change to send queue. In response to the first event, the change to send queue (first temporary storage location) is searched in order to identify the presence of the archive file, and its size/type prior to being sent to the OneDrive cloud server. Http://go.microsoft.com/fwlink/p/?linkid=829044.
(d) the step of transmitting the archive file to a second storage location responsive to a second event, the second storage location being an intermediate or a permanent storage location,	When the identified archived file has a larger size, then it is divided into chunks for uploading. After dividing the file into smaller chunks (if needed), one by one, these chunks are transmitted from the change to send queue (first temporary storage) to the OneDrive cloud server (permanent storage) in

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

wherein the first event is different from the second event.	response to a different second event that either indicates the OneDrive server is ready to receive data, or indicates which blocks to send. After OneDrive determines (second event) which files or blocks (archive file) are different than what it already has saved, the archive file is then transmitted to an output buffer, or a local cache, and external cache, or an intermediary server (intermediate storage location), or to the OneDrive server (permanent storage) in response to the second event.
Claim 29	
A *non-transitory computer readable medium comprising program instruction for performing the method of claim 15.*	Defendant hosts a computer server storing a computer executable application (called OneDrive) that has computer readable code that performs the method steps of claim 15, as detailed above (https://www.microsoft.com/en-us/microsoft-365/onedrive/download).

87. As described above, Defendant's products or services infringe the '348 Patent.

88. Defendant's OneDrive service includes a program which can be installed on a computer or computing device.

89. Defendant's OneDrive program detects instructions to perform operation(s) files.

90. When an instruction is detected, Defendant's OneDrive program creates an archived version of the file being modified and stores the archives file in a temporary storage location.

91. Later, the archived file is moved to another storage location.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

92. Defendant has actual knowledge of Plaintiff's rights in the '348 Patent and details of OneDrive's infringement of the '348 Patent based on at least the filing and service of this Complaint.

93. By the foregoing acts, including its testing and use of OneDrive, Defendant has directly infringed, literally and/or under the doctrine of equivalents, the '348 Patent in violation of 35 U.S.C. § 271.

94. Defendant has knowledge of the '348 Patent and indirectly infringes the '348 Patent by active inducement under 35 U.S.C. § 271(b) and/or § 271(f). Defendant has induced, caused, urged, encouraged, aided and abetted its direct and indirect customers to make, test, use, sell, offer for sale and/or import accused products and/or services. Defendant has done so by acts including but not limited to selling accused products and/or services to its customers; marketing accused products and/or services; and providing instructions, technical support, and other support and encouragement (available at https://support.microsoft.com/en-us/office/back-up-your-documents-pictures-and-desktop-folders-with-onedrive-d61a7930-a6fb-4b95-b28a-6552e77c3057, or https://support.microsoft.com/en-US/search/results?query=restore+file+from+backup+OneDrive&isEnrichedQuery=true, for instance) for the use of accused products and/or services. Such conduct by Defendant was intended to and actually resulted in direct infringement, including the making, testing, using, selling, offering for sale, and/or importation of accused products and/or services in the United States.

95. Defendant has knowledge of the '348 Patent and indirectly infringes by contributing to the infringement of, and continuing to contribute to the infringement of, one or more claims of the '348 Patent under 35 U.S.C. § 271(c) and/or 271(f) by selling, offering for sale, and/or importing into the United States, the accused products and/or services. Defendant knows that the accused products and/or services include hardware components and software instructions that work in concert to perform specific, intended functions. Such specific, intended

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

functions, carried out by these hardware and software combinations, are a material part of the inventions of the '348 Patent and are not staple articles of commerce suitable for substantial non-infringing use.

96. Upon information and belief, Defendant's infringement was knowing, intentional and willful.

97. Defendant's acts of infringement of the '348 Patent has caused Plaintiff damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284.

98. Upon information and belief, Defendant has gained profits by virtue of its infringement of the '348 Patent.

99. The circumstances surrounding Defendant's infringement are exceptional and, therefore, Plaintiff is entitled to an award of attorney's fees pursuant to 35 U.S.C. § 285.

THIRD CLAIM FOR RELIEF

(Infringement of the '850 Patent – 35 U.S.C. §271)

100. Plaintiff hereby incorporates by reference and realleges the foregoing paragraphs as if fully set forth herein.

101. Plaintiff is the current exclusive owner and assignee of all right, title, and interest in and to the '850 Patent titled "Automatic Real-Time File Management Method and Apparatus".

102. The '850 Patent was duly and legally issued by the United States Patent and Trademark Office on March 10, 2020.

103. Plaintiff owns all rights to the '850 Patent including the right to bring this suit for damages.

104. A true and correct copy of the '850 patent is attached hereto as Exhibit C.

105. The '850 Patent is valid and enforceable.

106. Defendant has directly and indirectly infringed the '850 patent by making, using, testing, selling, offering for sale, and/or importing into the United States, without authority, products, methods performed by and/or attributable to equipment, and or services that practice

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

one or more claims of the '850 Patent, including but not limited to its OneDrive file hosting product and services.

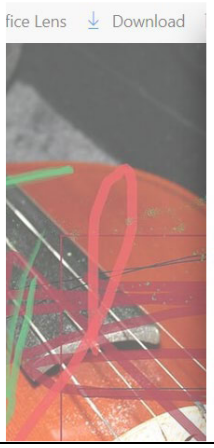
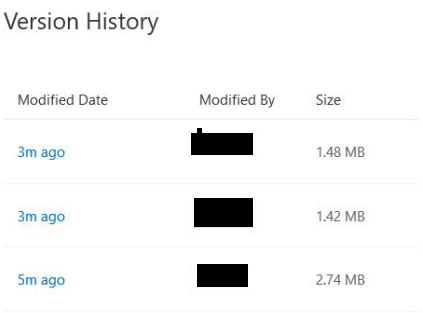
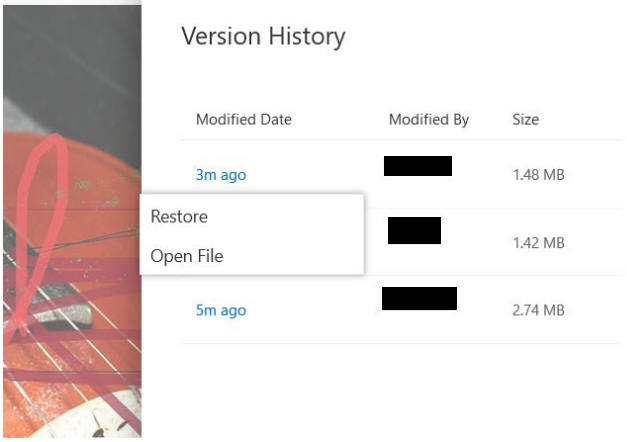
107. As a non-limiting example, Defendant has infringed Claims 1 – 18 of the '850 Patent. Defendant, without authorization from Plaintiff, has marketed, used, tested, offered for sale, sold, and/or imported into the U.S., including within this District, its OneDrive file hosting service that infringes at least Claim 1 – 21 of the '850 Patent.

108. As a non-limiting example, set forth below (with claim language in italics) is a description of infringement of exemplary Claim 10 of the '850 Patent in connection with OneDrive. This description is based on publicly available information. Plaintiff reserves the right to modify this description, including, for example, on the basis of information about OneDrive that it obtains during discovery.

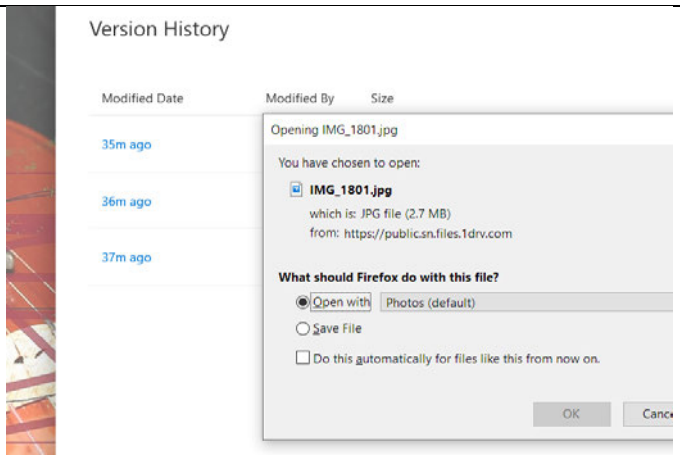
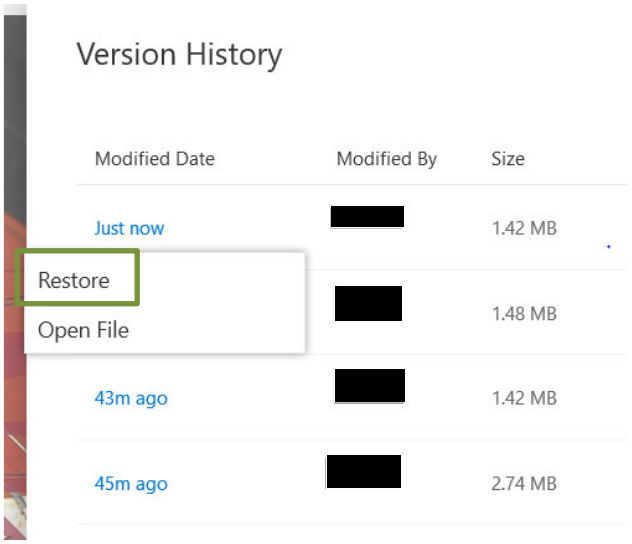
Claim 10

A method of restoring a file to a previous version of the file, a current version of the file being available at a local storage location, comprising the steps of:	OneDrive provides a method for users to restore a file to a previous version from the current version. The current version of the file is stored locally on the user's computer.
(A) presenting information for a collection of one or more previous versions of the file, the information for the collection including information indicative of at least one or more of previous versions of the file, wherein a restorable representation of each version, V, of the previous versions, is	OneDrive maintains a Version History of a given file. When a user selects to view the Version History of a document, OneDrive displays the previous versions of the file to the user via their web interface. Previous versions are stored remotely from the user's computer on OneDrive's server. Each previous version is available for selection for previewing and/or restoring the current

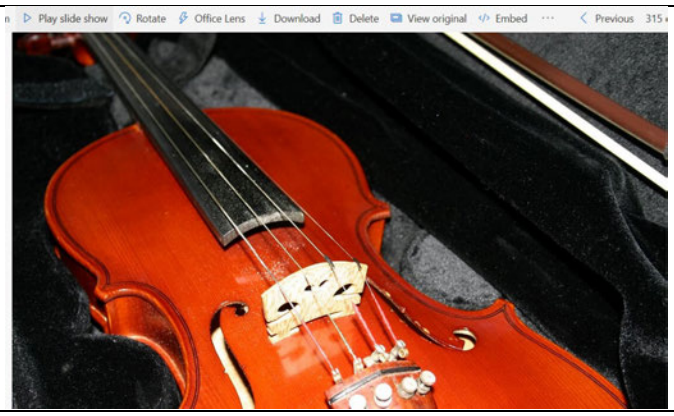
BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

retrievable from a remote storage location, the restorable representation having at least information required for recovering the version V, the remote storage location being accessible through a network;	version via the web interface. When selected, OneDrive retrieves the selected previous version from the remote storage and restores the current version of the file to its previous version.  
(B) responsive to a selection to preview a selected previous version of the file based on the presented information for the collection of (A), presenting a presentable representation of the selected previous version, the selected previous version being one of the previous versions of the file in the presented information for the collection, the presentable representation having at least information required for presenting at least a portion of the selected previous version;	For each previous version that is clicked on by the user, the user can open (download/open) the file to preview at least a portion of the previous version of the file, as shown below in screen shots. 

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

	
(C) responsive to a selection to restore the selected previous version, retrieving the restorable representation of the selected previous version from the remote storage location and storing the selected previous version as the current version on the local storage location, the selected previous version available from the restorable representation of the selected previous version.	The user can select which previous version it wishes to restore and clicks the restore button to indicate its selection of the previous version.  OneDrive then retrieves the selected previous version from the OneDrive server and restores the current file back to the particular selected previous version. The restored file is then stored locally on the user's computer.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

109. As described above, Defendant's products or services infringe the '850 Patent.

110. Defendant's OneDrive service allows for restoring a file to a previous version of the file.

111. Defendant's OneDrive service maintains current version of files at a local storage location.

112. Defendant's OneDrive service maintains information for previous versions of files.

113. Defendant's OneDrive service allows for retrieving a version of files from a remote storage location.

114. Defendant has actual knowledge of Plaintiff's rights in the '850 Patent and details of OneDrive's infringement of the '850 Patent based on at least the filing and service of this Complaint.

115. By the foregoing acts, including its testing and use of OneDrive, Defendant has directly infringed, literally and/or under the doctrine of equivalents, the '850 Patent in violation of 35 U.S.C. § 271.

116. Defendant has knowledge of the '850 Patent and indirectly infringes the '850 Patent by active inducement under 35 U.S.C. § 271(b) and/or § 271(f). Defendant has induced, caused, urged, encouraged, aided and abetted its direct and indirect customers to make, test, use, sell, offer for sale and/or import accused products and/or services. Defendant has done so by acts including but not limited to selling accused products and/or services to its customers; marketing

BYRNES ♦ KELLER ♦ CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

accused products and/or services; and providing instructions, technical support, and other support and encouragement (available at https://support.microsoft.com/en-us/office/back-up-your-documents-pictures-and-desktop-folders-with-onedrive-d61a7930-a6fb-4b95-b28a-6552e77c3057, or https://support.microsoft.com/en-US/search/results?query=restore+file+from+backup+OneDrive&isEnrichedQuery=true, for instance) for the use of accused products and/or services. Such conduct by Defendant was intended to and actually resulted in direct infringement, including the making, testing, using, selling, offering for sale, and/or importation of accused products and/or services in the United States.

117. Defendant has knowledge of the '850 Patent and indirectly infringes by contributing to the infringement of, and continuing to contribute to the infringement of, one or more claims of the '850 Patent under 35 U.S.C. § 271(c) and/or 271(f) by selling, offering for sale, and/or importing into the United States, the accused products and/or services. Defendant knows that the accused products and/or services include hardware components and software instructions that work in concert to perform specific, intended functions. Such specific, intended functions, carried out by these hardware and software combinations, are a material part of the inventions of the '850 Patent and are not staple articles of commerce suitable for substantial non-infringing use.

118. Upon information and belief, Defendant's infringement has, and continues to be, knowing, intentional and willful.

119. Defendant's acts of infringement of the '850 Patent has caused, and will continue to cause, Plaintiff damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284.

120. Upon information and belief, Defendant has gained profits by virtue of its infringement of the '850 Patent.

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

121. The circumstances surrounding Defendant's infringement are exceptional and, therefore, Plaintiff is entitled to an award of attorney's fees pursuant to 35 U.S.C. § 285.

DEMAND FOR JURY TRIAL

122. Pursuant to Federal Rule of Civil Procedure 38(b), Plaintiff respectfully demands a jury trial of all issues triable to a jury in this action.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff prays for judgment against Defendant as follows:

A. A judgment and order adjudicating and declaring that Defendant has directly infringed the '478 Patent, literally or under the doctrine of equivalents;

B. A judgment and order adjudicating and declaring that Defendant has indirectly infringed the '478 Patent by inducing and/or contributing to its customers' infringement.

C. A judgment and order adjudicating and declaring that Defendant has directly infringed the '348 Patent, literally or under the doctrine of equivalents;

D. A judgment and order adjudicating and declaring that Defendant has indirectly infringed the '348 Patent by inducing and/or contributing to its customers' infringement.

E. A judgment and order adjudicating and declaring that Defendant has directly infringed the '850 Patent, literally or under the doctrine of equivalents;

F. A judgment and order adjudicating and declaring that Defendant has indirectly infringed the '850 Patent by inducing and/or contributing to its customers' infringement.

G. A judgment and order that Defendant must account for and pay actual damages (but no less than a reasonable royalty), to Plaintiff for Defendant's infringement of the '478 Patent, the '348 Patent and/or the '850 Patent, including supplemental damages for any continuing post-verdict infringement up until entry of the final

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

judgment and an award of an ongoing royalty for Defendant's post-judgment infringement in an amount according to proof in the event that a permanent injunction preventing future acts of infringement is not granted;

H. A determination that this case is exceptional under 35 U.S.C. § 285;

I. A judgment and order awarding Plaintiff its reasonable attorneys' fees;

J. A judgment and order awarding Plaintiff its costs, expenses, and interest, including pre-judgment and post-judgment, as provided for by 35 U.S.C. § 284;

K. A judgment and order that Defendant's infringement has been willful and an award of treble damages pursuant to 35 U.S.C. § 284;

L. A judgment and order awarding pre-judgment and post-judgment interest on each and every monetary award; and

M. Granting Plaintiff any such other and further relief as this Court deems just and proper, or that Plaintiff may be entitled to as a matter of law or equity.

DATED this 31st day of October, 2022.

BYRNES KELLER CROMWELL LLP

By /s/ Bradley S. Keller
Bradley S. Keller, WSBA #10665

By /s/ Jofrey M. McWilliam
Jofrey M. McWilliam, WSBA #28441
1000 Second Avenue, 38th Floor
Seattle, Washington 98104
206-622-2000
bkeller@byrneskeller.com
jmcwilliam@byrneskeller.com

Gregory P. Sitrick (*pro hac vice* forthcoming)
Isaac S. Crum (*pro hac vice* forthcoming)
MESSNER REEVES LLP
7250 N. 16th Street, Suite 410
Phoenix, AZ 85020
gsitrick@messner.com
icrum@messner.com
Attorneys for Plaintiff Datanet LLC

BYRNES • KELLER • CROMWELL LLP
38TH FLOOR
1000 SECOND AVENUE
SEATTLE, WASHINGTON 98104
(206) 622-2000

EXHIBIT A

US008473478B2

(12) **United States Patent**
Roach et al.

(10) **Patent No.: US 8,473,478 B2**
(45) **Date of Patent: Jun. 25, 2013**

(54) **AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS**

(76) Inventors: **Warren Roach**, Colorado Springs, CO (US); **Steven R. Williams**, Colorado Springs, CO (US); **Troy J. Reiber**, Colorado Springs, CO (US); **Steven C. Burdine**, Middleton, WI (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1836 days.

(21) Appl. No.: **09/957,459**

(22) Filed: **Sep. 21, 2001**

(65) **Prior Publication Data**

US 2002/0065837 A1 May 30, 2002

Related U.S. Application Data

(60) Provisional application No. 60/234,221, filed on Sep. 21, 2000.

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.**
USPC **707/707**

(58) **Field of Classification Search**
USPC 707/203, 204, 200, 201, 102, 202
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,276,867	A	1/1994	Kenley et al.	395/600
5,638,059	A *	6/1997	Pilkington	340/970
5,668,991	A	9/1997	Dunn et al.	395/618
5,751,997	A	5/1998	Kullcik et al.	395/489
5,765,173	A	6/1998	Cane et al.	707/204
5,978,815	A	11/1999	Cabrera et al.	707/204
6,023,710	A	2/2000	Steiner et al.	707/204
6,047,294	A	4/2000	Deshayes et al.	707/204
6,101,507	A	8/2000	Cane et al.	707/204
6,460,055	B1 *	10/2002	Midgley et al.	707/204
6,535,894	B1 *	3/2003	Schmidt et al.	707/204
6,629,109	B1 *	9/2003	Koshisaka	707/203
7,117,371	B1 *	10/2006	Parthasarathy et al.	713/187

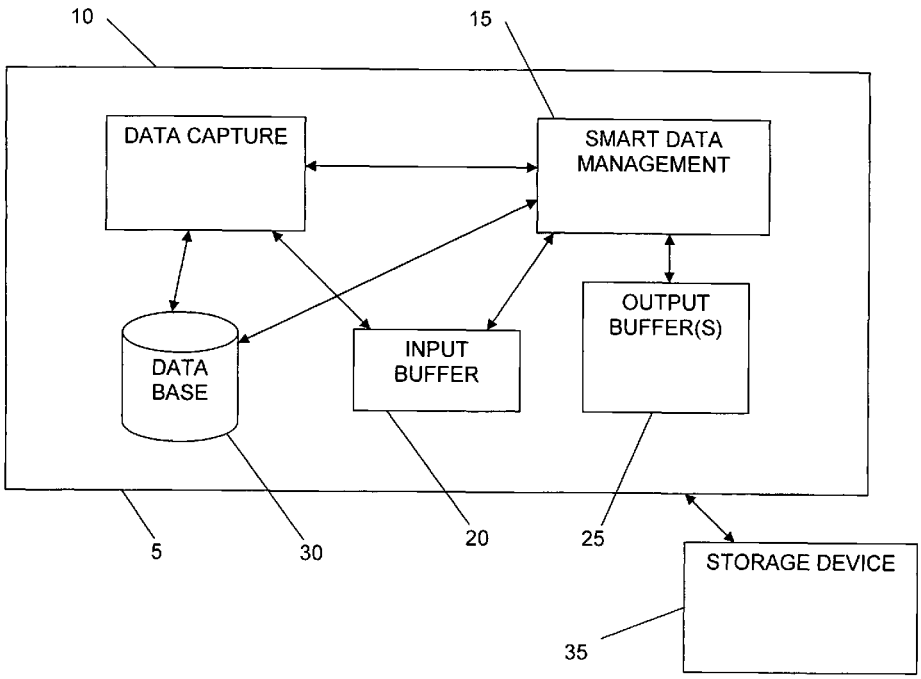
* cited by examiner

Primary Examiner — Baoquoc N To
(74) *Attorney, Agent, or Firm* — Cahn & Samuels, LLP

(57) **ABSTRACT**

A method for archiving files includes determining when a change in an operating file is imminent, capturing the operating file immediately before the change in the operating file occurs, if the operating file has not already been captured; and capturing the operating file immediately after the change in the operating file has occurred.

11 Claims, 6 Drawing Sheets



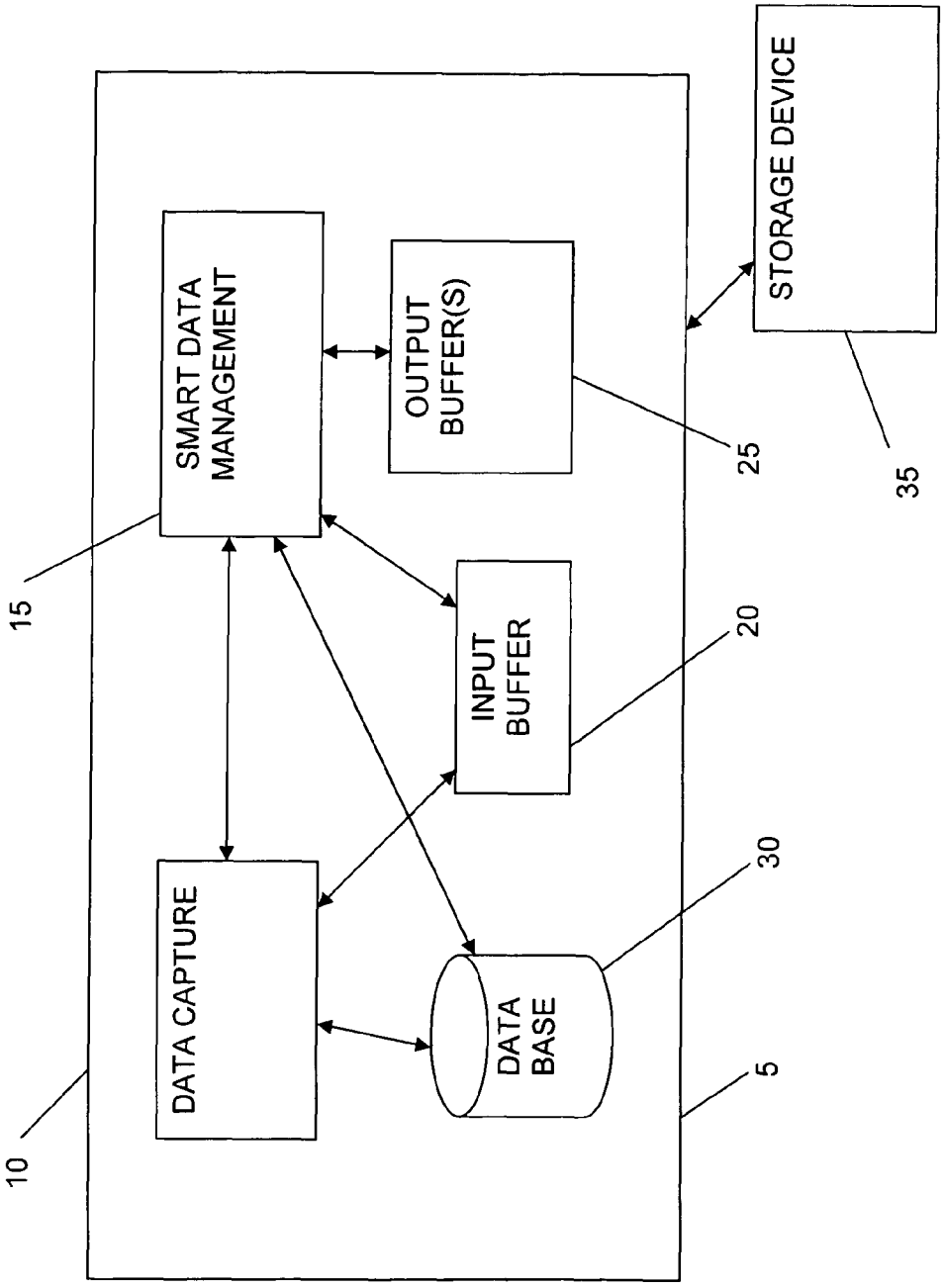

STORAGE DEVICE
SMART DATA MANAGEMENT
OUTPUT BUFFER(S)
INPUT BUFFER
DATA CAPTURE
DATA BASE
10
15
20
25
30
35
5

FIGURE 1

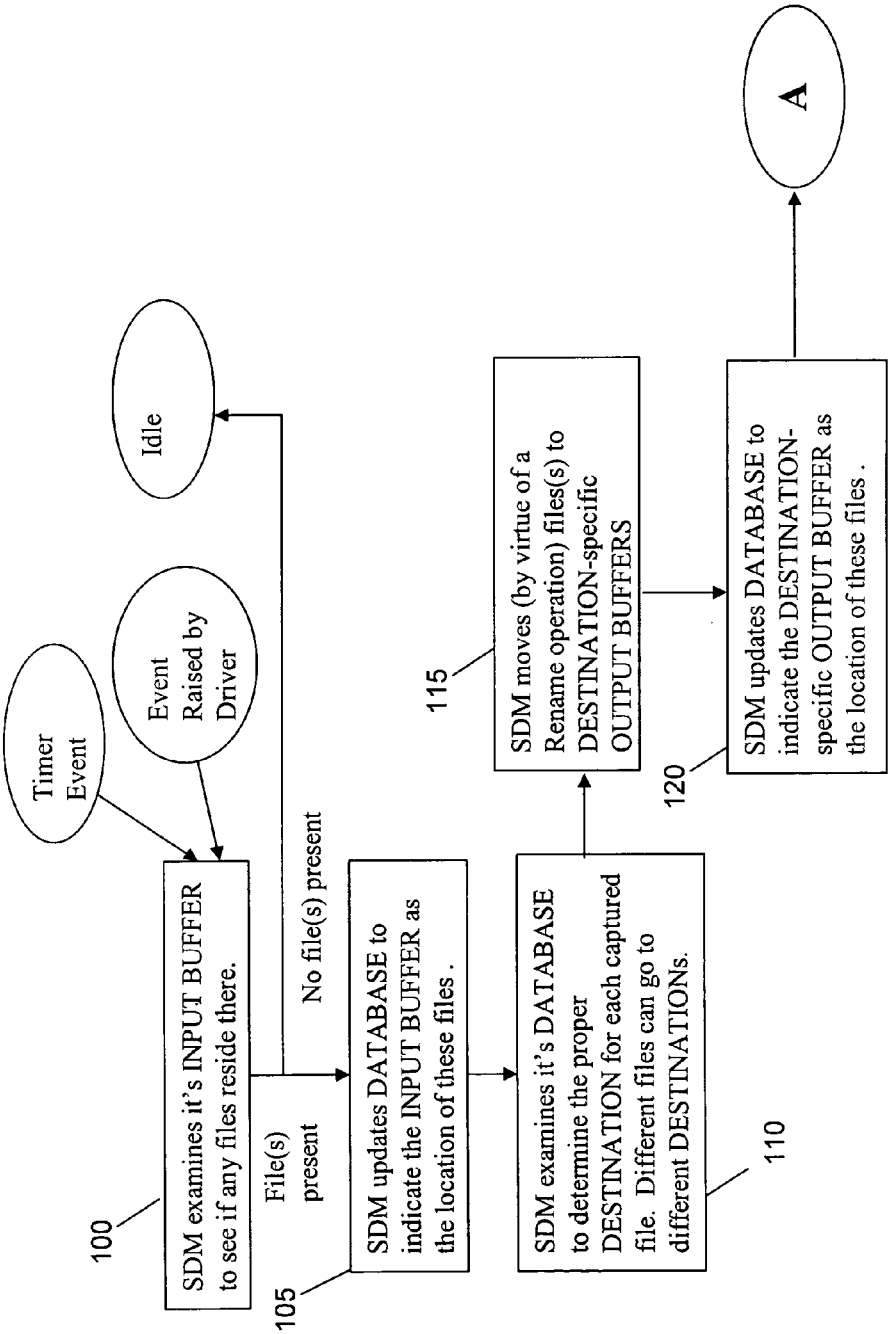

Timer Event
Event Raised by Driver
Idle
100
SDM examines it's INPUT BUFFER to see if any files reside there.
File(s) present
No file(s) present
105
SDM updates DATABASE to indicate the INPUT BUFFER as the location of these files .
110
SDM examines it's DATABASE to determine the proper DESTINATION for each captured file. Different files can go to different DESTINATIONs.
115
SDM moves (by virtue of a Rename operation) files(s) to DESTINATION-specific OUTPUT BUFFERS
120
SDM updates DATABASE to indicate the DESTINATION-specific OUTPUT BUFFER as the location of these files .
A

FIGURE 2A

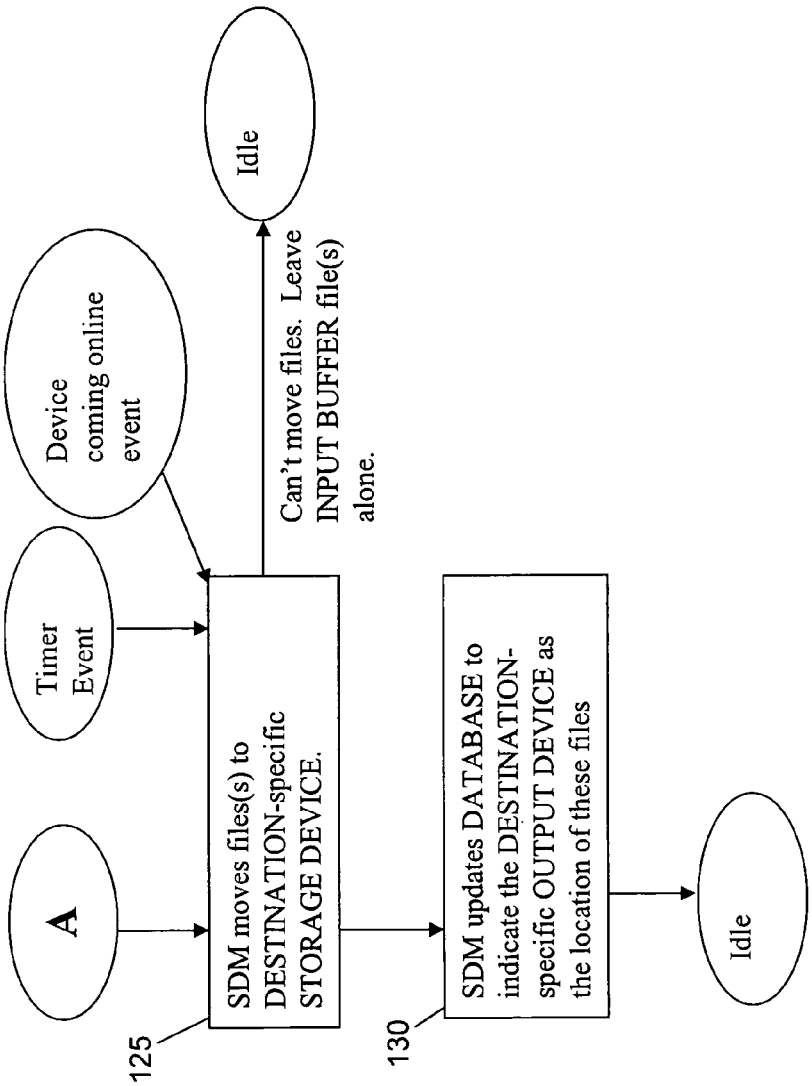

A
Timer Event
Device coming online event
SDM moves files(s) to DESTINATION-specific STORAGE DEVICE.
125
Can't move files. Leave INPUT BUFFER file(s) alone.
Idle
SDM updates DATABASE to indicate the DESTINATION-specific OUTPUT DEVICE as the location of these files
130
Idle

FIGURE 2B

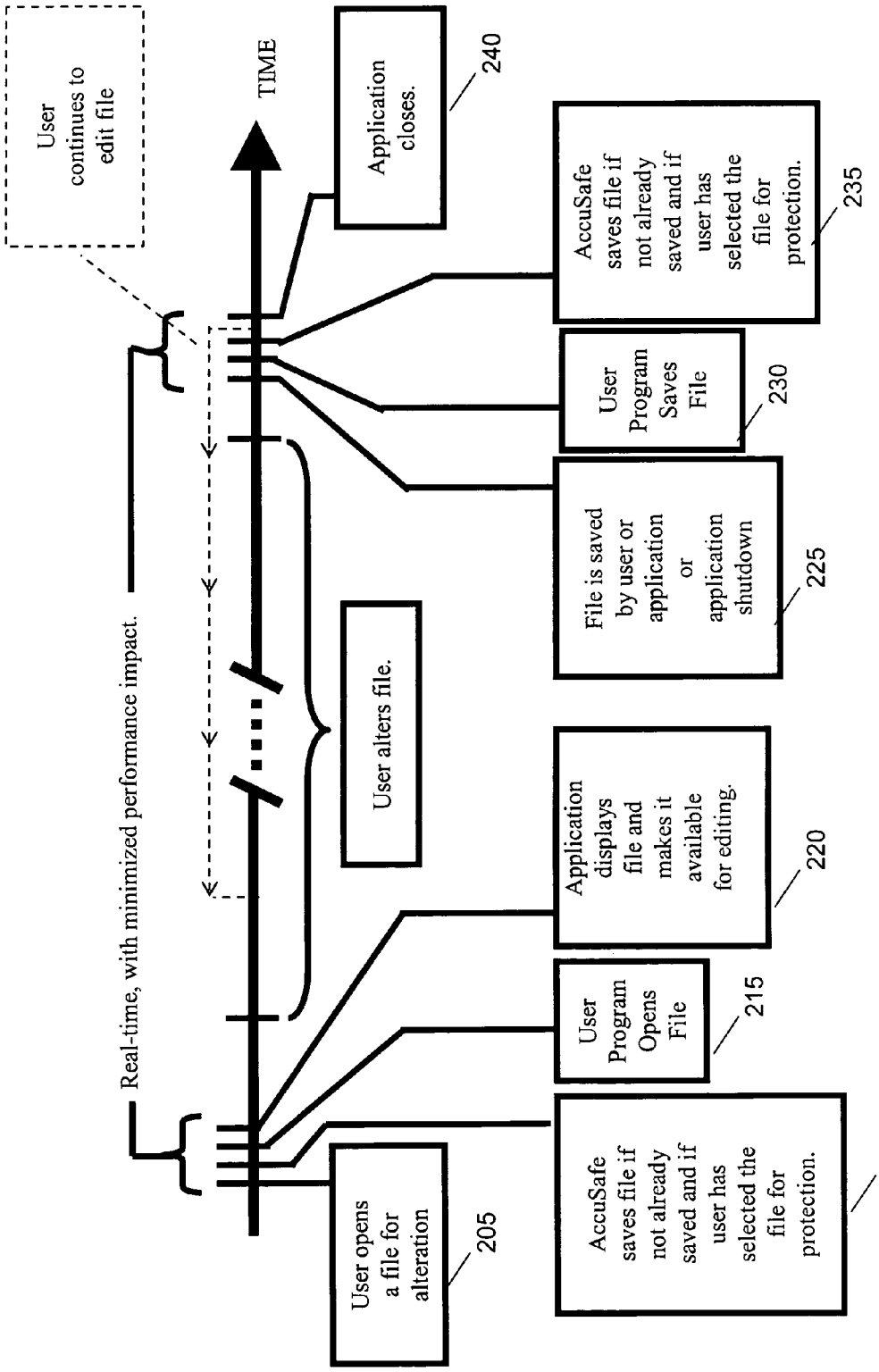

User continues to edit file
TIME
Application closes.
240
AccuSafe saves file if not already saved and if user has selected the file for protection.
235
User Program Saves File
230
File is saved by user or application or application shutdown
225
User alters file.
Application displays file and makes it available for editing.
220
User Program Opens File
215
AccuSafe saves file if not already saved and if user has selected the file for protection.
210
User opens a file for alteration
205
Real-time, with minimized performance impact.

FIGURE 3

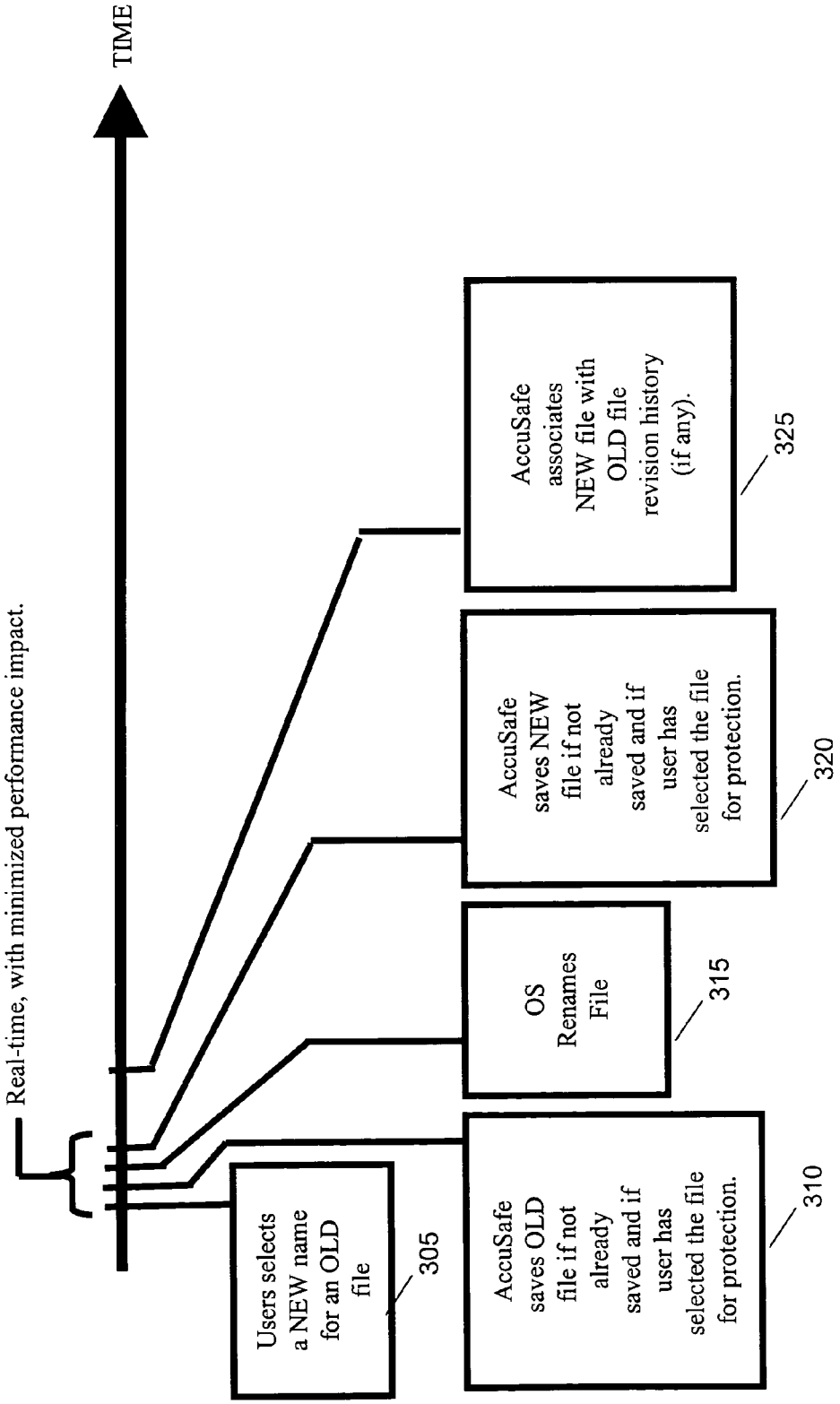

TIME
Real-time, with minimized performance impact.
Users selects a NEW name for an OLD file
305
AccuSafe saves OLD file if not already saved and if user has selected the file for protection.
310
OS Renames File
315
AccuSafe saves NEW file if not already saved and if user has selected the file for protection.
320
AccuSafe associates NEW file with OLD file revision history (if any).
325

FIGURE 4

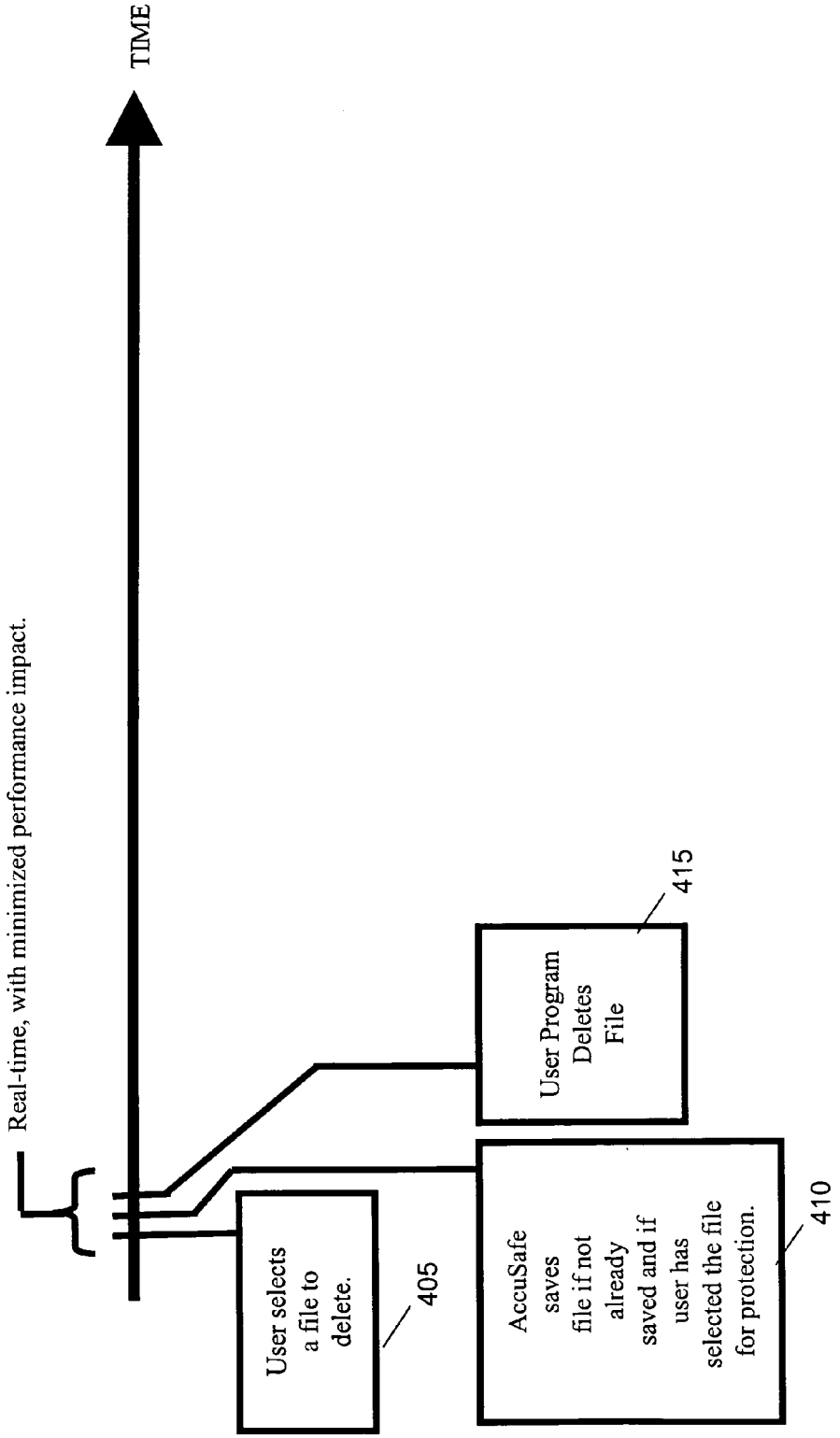

TIME
Real-time, with minimized performance impact.
User selects a file to delete.
405
AccuSafe saves file if not already saved and if user has selected the file for protection.
410
User Program Deletes File
415

FIGURE 5

AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS

This application claims the benefit of Provisional Application No. 60/234,221 filed Sep. 21, 2000, which is herein incorporated by reference.

FIELD OF THE INVENTION

This invention relates to file preservation, file capture, file management and file integrity techniques for computing devices.

BACKGROUND OF THE INVENTION

One of the greatest challenges faced by information technology (IT) professionals and computer users today, particularly in the business environment is the protection and management of data. Data may be stored on user workstations, e.g., laptop computers, home or office desktop computers, network servers or other devices external to the workstations. Important data may even be stored on hand-held computing devices such as PDAs, PALs and other like devices. Complicating the problem is the fact that the criticality of data is increasing and the difficulty of managing it, protecting it from loss and keeping it available is increasing. This is due to a variety of factors, including: 1) the explosion in data volume, particularly that stored on desktop and laptop computers, 2) the increasing complexity of desktop and laptop computer software and hardware and increasing trends toward a paperless environment were absolute reliance (because paper copies are becoming less the norm) on data integrity is increasingly significant.

Many home computer users do not realize the vulnerability of their computer data. Many that do understand the very real potential for data loss, purchase backup systems whose operation and user interface is often confusing and/or time-consuming to use, dramatically decreasing their effectiveness or dependability. As a result, many computer users remain very much at risk of data loss resulting from hardware and/or software failures, fires, stolen equipment, etc. While these risks are significant, the most frequent cause of data loss is user error (accidental file deletes, file overwrites, errant programs, etc.), to which users remain very vulnerable even with most present day backup systems.

The financial impact of information loss is substantial. As reported by the Safeware Insurance Agency, in 1999 alone, insurance claims for damaged, lost and stolen computers (primarily notebook computers) totaled more than $1.9 billion. This figure does not include the untold billions lost in intellectual capital and time. It is costly to recreate lost data and there are significant related costs such as lost productivity and lost opportunity. Consider, for example, the financial and health related impact of a doctor losing all patient contact information and medical histories due to a hard disk crash or some other type of computer failure. In addition, it is costly to keep desktop and laptop computers up and running in the wake of their increasing complexity.

A variety of products have been developed to address data preservation and integrity issues. These products may be loosely grouped into three categories, manual backup systems, schedule based backup systems and mirroring backup systems.

The least efficient and probably one of the most frequently used backup systems is the manual backup. At times determined by the user, the user selects files to be backed up and either utilizes the built in backup procedure for the corresponding application or manually copies the selected files to a desired backup storage media.

The problems with this method of preserving data are self-evident. Backup procedures are often confusing and may differ from application to application. Accordingly, the user must familiarize itself with the various methods for performing backups. In addition, users may forget to backup or elect not to on a given occasion due to time constraints or other reasons. Manual backups often do not allow the user to continue to use the system during the backup procedure. Furthermore, data stored to the backup media is really only a "snapshot" of the data at the time that the backup is performed. Any changes made between manual backups would be lost if there was a failure on the computer's storage device.

Schedule based backup systems typically perform backups according to a schedule either set by the user or preset by the backup software. One of the major disadvantages of each type of schedule-based backup system is that as with manual backups, they miss work done between schedule points. This may cause the user to loose critical information as they work between schedule points. Another disadvantage of schedule-based backups systems is that they are frequently confusing and cumbersome for the user. Still another disadvantage of schedule-based backup systems is that they function poorly if at all when the backup storage device is unavailable, i.e., they cannot be written to due to a communications error or because the device has reached its capacity, is bandwidth limited, or is non-operational for some other reason.

Mirroring is a technique typically applied to disk based backup systems. Mirroring backup systems are the most comprehensive in that everything that happens to the source storage device immediately happens to the backup storage device. That is the backup drive becomes a mirror image of the source drive. Accordingly, if a failure occurs on the source disk, processing can be switched to the backup disk with little or no service interruption.

The strongest advantage of mirroring systems is also their strongest disadvantage. Because there is no operational discrimination, if a file is accidentally deleted from the source disk, it is deleted and cannot be preserved on the backup disk. Likewise, if a virus infects the source disk it is likely to infect the backup disk. Another disadvantage of mirroring systems is that separate backup disks are required for each source disk, doubling the disk requirement for the system. The backup disk must be at least as large as the source disk and the disks must be configured with identical volume mapping. Any extra space that may be present on the backup disk is unavailable.

All of these methods require that the user specify which files/directories to back up, but many users have no concept of files and directories in their thought process, much less are they able to correlate a particular application (e.g. Microsoft Excel) with the kinds and locations of files they generate. These systems simply require too much user knowledge, and too much user intervention. The backup user's risk increases dramatically the lower his computer knowledge may be.

In view of the foregoing, there is a need for a file capture, preservation and management system that captures files just before and/or just after they have been changed to minimize loss of data between backup events. There is also a need for file capture and preservation system that captures files even when the destination storage medium for the files is unavailable. There is a further need for a system that allows users to recover easily and quickly from any type of information loss, including simple user errors, failed software installations or updates, hardware failures (attached storage devices), and lost or stolen laptop computers. Users should be able to recover on their own, without the intervention of the IT staff,

and their backup systems should be as "behind the scenes" as possible, requiring little user attention and extremely small amounts of user computer knowledge.

SUMMARY OF THE INVENTION

It is an object of the invention to a file capture, preservation and management method and apparatus that captures files just before and/or just after the files are changed.

It is another object of the invention to provide a file capture, preservation and management method and apparatus that has an imperceptible impact on system performance from the user's point of view.

It is a further object of the invention to provide a file capture, preservation and management method and apparatus that captures and stores files even when there is no connection to the desired storage location.

Still another object of the invention is to provide a file capture, preservation and management method and apparatus that captures and stores files even when the desired storage location is unavailable.

In accordance with an aspect of the invention, a method for archiving files is provided. The method includes, in a computing device, detecting an instruction from a resident program to perform an operation on an operating file. Upon detection of the instruction, capturing the operating file temporally proximate to the operation being performed on the operating file.

In accordance with another aspect of the invention, a method for moving files from a first storage location to a second storage location is provided. The method includes, in a computing device, searching a first storage location for files responsive to the occurrence of a first event and moving the files from the first storage location to the second storage location responsive to a second event.

In accordance with still another aspect of the invention, a method for archiving files is provided. The method includes detecting an instruction from a resident program to perform an operation on an operating file. The method further includes creating an archive file from the operating file and storing the archive file in a first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction. In keeping with the method, the first storage location is searched for an archive file responsive to the occurrence of a first event. The archive file is then moved from the first storage location to the second storage location responsive to a second event.

The accompanying figures show illustrative embodiments of the invention from which these and other of the objectives, novel features and advantages will be readily apparent.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a block diagram of a computing device in accordance with the present invention.

FIG. **2**A is a flow chart depicting a process for moving files in accordance with the present invention.

FIG. **2**B is a flow chart showing another process for moving files in accordance with the invention.

FIG. **3** is a time line illustrating a sequence of events in an exemplary operation in accordance with the invention.

FIG. **4** is a time line illustrating a sequence of events in another exemplary operation in accordance with the invention.

FIG. **5** is a time line illustrating a sequence of events in still another exemplary operation in accordance with the invention.

DETAILED DESCRIPTION OF THE EMBODIMENT

Definitions

Operating System (OS)—A computer program that allocates system resources such as memory, disk space, and processor usage and makes it possible for the computer to boot up to a human user interface allowing the user to interact with the computer and control its operation.

Operating File—a system or user file.

Archive File—a file containing all of the data of an operating file in a native or altered format and/or a file containing at least some of the data of an operating file and including references to the location of the remainder of the data of the operating file.

Computing Device—a personal computer, a laptop or notebook computer, a server, a hand-held computing device, a PDA or a PAL. The term computing device is not specific to the kind of operating system being run on such computing device, and includes devices running Microsoft operating systems, Apple Macintosh operating systems, UNIX operating systems, Linux operating systems, and other operating systems.

Storage Location—any storage device, or a buffer, folder, directory or designated area on a storage device.

Personal Attached Storage Device—any internal or external storage device connected to a computing device.

Network Attached Storage Device—any storage device connected directly to a network to which a first computing device is also temporarily or permanently connected, or any storage device connected to a second computing device that is also temporarily or permanently connected to the network to which the first computing device is temporarily or permanently connected.

Internet storage area network—any storage area (device, collection of devices, etc.) that can be accessed by the computing device when the computing device is temporarily or permanently connected to the Internet.

Peer-to-Peer Storage Device—any storage area (device, collection of devices, etc.) that can be accessed by the computing device when it is sharing resources with other network or Internet accessible computers.

Resident Program—an operating system (OS) or other program that has control over file operations such as "read", "write", "save", "rename", "delete", "copy", "move", "open", "close", etc.

User Program—an application software program or other computing program installed by the user or by the computer manufacturer for user creation of desired data, documents, or other information that is designed to enhance the functionality and/or enjoyment and/or usability of the computing device. The present invention is directed to an apparatus and/or method for file capture, preservation and management. The invention includes a file capture aspect and smart data management aspect. The invention may be realized as a method and/or an apparatus. More particularly, the invention may be realized as a set of program code instructions stored on a computer usable medium, a set of program code instructions embodied in a signal for transmitting computer information, and a processor and/or computing device configured as described herein.

FIG. **1** depicts a block diagram in accordance with the present invention comprising a computing device **5** including a file capture block **10** (or file capturer), a smart data management block **15** (or smart data manager), an input buffer **20**, output buffer(s) **25**, and a database **30**. A storage device **35** is

also provided and may be either internal or external to computing device **5**. The invention functions in conjunction with a resident program on computing device **5**.

In accordance with an embodiment of the invention, file capture block **10** detects an instruction to perform an operation on an operating file initiated by the resident program of computing device **5**. At a moment temporally proximate to when the resident program actually performs the operation, i.e., just before and/or just after the operation is performed on the operating file, or, more preferably, the instant before and/or the instant after the operating file is changed, file capture block **10** captures the operating file or portions thereof. Preferably, the operating file is captured within a few clock cycles of the detection of the instruction.

In keeping with a preferred aspect of the invention, file capture block **10** causes the location of the captured operating file to be recorded in database **30**. The continued process of recording information about captured operating files, or portions thereof, in database **30** creates a record of each version of the operating file, which may be accessed by the user or by other programs.

File capture is preferably executed by creating an archive file from the operating file. The archive file is preferably stored in a temporary storage location, internal or external to the computer, such as input buffer **20**. However, the archive file may be stored directly in storage device **35**. In accordance with a preferred aspect of the invention, storage device **35** may be a personal attached storage device, a network attached storage device, an Internet storage area network, a peer-to-peer storage device, or other storage device.

In keeping with a preferred aspect of the invention, smart data management block **15** manages the migration of the archive file from the input buffer **20** through the output buffers **25** to storage device **35**. This migration may take place either synchronously or asynchronously with the file capture procedures described herein. The time duration from a file arriving in input buffer **20** and when it arrives on archive storage device **35** is managed by the smart data management block **15**. More particularly smart data management block **15** regularly examines input buffer **20** for the presence of archive files. Smart data management block **15** performs this examination upon the occurrence of an event, e.g., messages from the file capture block **10** and/or various messages from the resident program(s), messages from an input buffer timer sent at time intervals controlled by a timer or at time intervals selected by the user. Optionally, smart data management block **15** may then examine database **30** to determine a defined storage location for each of the archive files stored in input buffer **20**. Each archive file stored in the input buffer **20** may be directed to the same storage location or to different storage locations and archive files may be directed to multiple storage locations for redundancy. Preferably, smart data management block **15** moves the archive files to one or more output buffers **25**. More preferably each archive file is moved to output buffer(s) **25** corresponding to the final storage location(s) for that archive file. Alternatively, all archive files may be moved to a single common output buffer **25** if desired. Upon the occurrence of an event, and/or at defined time intervals, smart data management block **15** moves the archive files from the output buffers **25** to their respective storage device(s) **35**. Exemplary events include but are not limited to messages indicating when storage device **35** is connected and ready for use, messages indicating when storage device **35** is inserted/removed, full, defective, etc., and messages indicating when storage device **35** is disconnected or unavailable, and messages from a storage device timer sent at time intervals controlled by the timer or at time intervals controlled by the user. The input buffer timer and the storage device timer may operate synchronously or non-synchronously.

Under certain conditions, smart data management block **15** may be unable, or may elect not to move the archive files. For example, if storage device **35** is unavailable then smart data management block **15** will not move the archive files to storage device **35**. Among the conditions that may cause storage device **35** to be unavailable are i) storage device **35** is disconnected from computing device **5**, ii) the connection between storage device **35** and computing device **5** is faulty or unacceptably slow, iii) storage device **35** is full, or iv) storage device **35** is malfunctioning. In addition, smart data management block **15** may also regulate movement of archive files according to time schedules set by the user, by monitoring connection bandwidth availability and moving files only during times of high bandwidth availability, or by monitoring other factors including messages that may received from storage location server requests for archive file transmittal.

A preferred operational mode for smart data management block **15** is illustrated in the flowcharts of FIGS. **2**A and **2**B. In step **100** of FIG. **2**A, smart data manager **15** examines input buffer **20** to determine whether any archive files are stored therein. If no archive files are present, smart data manager **15** rests idle until the next event occurs. If archive files are detected, in step **105**, smart data manager **15** updates database **25** to indicate the location of the archive files; that is, to indicate that the archive files are resident in input buffer **20**. In step **110**, smart data manager **15** examines database **30** to determine the proper destination for each archive file. In step **115**, smart data manager **15** moves the archive files to output buffers **25**. In step **120**, smart data manager **15** updates database **30** to indicate that the archive files are now stored in the output buffer.

In FIG. **2**B in step **125**, the archive files are moved to one or more storage devices **30**. If smart data manager **15** is unable to move the archive files to any of the storage devices **30**, smart data manager **15** rests idle and does not move the archive files until it is notified that the storage device is available. Accordingly, the archive files remain in either input buffer **20** or output buffer **25** until smart data management block **15** is notified. In step **130** smart data manager **15** updates database **25** to indicate that the archive files are stored in one or more storage devices **30**.

Use Specific to User Program Operations

The following examples are directed to embodiments of the invention specific to operations performed by a user program. The file capture, preservation and management processes of the invention are not limited to execution with the exemplary operation discussed below. The processes of the invention are preferably executed when a resident program causes a change or a change to be imminent in the operating file. Therefore, the following examples are intended to be exemplary only and nonlimiting.

File Capture at File Open

As illustrated in FIG. **3**, in step **205**, the user or a program selects an "open" operation to open an operating file and an instruction to perform that "open" operation on the operating file is sent to the resident program. In step **210**, file capture block **10** detects the instruction and captures the operating file. Optionally, prior to capturing the operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** creates an archive file and stores the archive file in a storage location such as input buffer **20** or storage device **35** just before the

resident program opens the operating file. Preferably, file capture block **10** stores the archive file in input buffer **20**. In step **215** the resident program opens the operating file and in step **220** the user program displays the operating file as originally requested, e.g. Microsoft Word, and makes it available for the user to alter, e.g., edit a word processing document, amend or add to a database, etc. Step **210** is performed by momentarily delaying the execution of step **215** in such a manner as to have little or no perceptible impact on system performance from the user's point of view.

In step **225**, the user program begins a process to save the altered operating file and an instruction to save the altered operating file is sent to the resident program. In step **230** the resident program saves the altered operating file pursuant to the instruction. In step **235**, immediately after the altered operating file is saved by the resident program, file capture block **10** captures the altered operating file, preferably by creating and storing an archive file of the altered operating file in input buffer **20**. In accordance with a preferred feature of the invention, file capture block **10** may save the archive file in such a way that previous revisions of the operating file are retained. That is, every time the operating file is changed, file capture block **10** saves an archive file and database **30** is updated with information about the archive file. Accordingly, over time, a plurality of archive files may be created from the original operating file. Each archive file represents a revision of the original operating file.

File Capture in the "RENAME" Operation

As illustrated in FIG. **4**, step **305**, in performing an operating file rename operation, the user or a program generates an instruction for the resident program to select a new name for an old operating file. In step **310**, file capture block **10** detects the instruction and captures the old operating file. Optionally, prior to capturing the old operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** creates an archive file of the old operating file and stores the archive file in a storage location such as storage device **35** or, more preferably, input buffer **20** just before the resident program renames the old operating file. In step **315** the resident program renames the old operating file, thus creating a new operating file. Immediately after the old operating file is renamed, file capture block **10** captures the new operating file. Optionally, prior to capturing the new operating file, file capture block **10** may determine whether the new operating file has previously been archived, whether the user has selected the new operating file for protection, or other matching conditions exist. Like the archive file for the old operating file, the archive file for the new operating file is preferably stored in input buffer **20**. In step **325** file capture block **10** and smart data management block **15** associate or link the new operating file with each of the versions of the old operating file to create a continuous operating file revision history.

File Capture in the "Delete" Operation

FIG. **5** illustrates the file capture process in the delete operation. In step **405**, the user or a program identifies an operating file to delete and generates an instruction to the resident program. In step **410**, file capture block **10** detects the instruction and captures the operating file just before it is deleted in step **415**. Optionally, prior to capturing the operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** preferably captures the operating file. In step **420**, the resident program deletes the operating file.

As shown by the examples given, a clear advantage of the invention is, regardless of the operation being performed, after each file capture step, file capture block **10** preferably updates database **30** to indicate the location of the corresponding archive file. Database **30** may keep track of multiple versions of an operating file, any of which may be accessed at the request of the user or other program.

Another advantage of the invention is that by capturing the operating file just before and/or just after an operation is performed thereon, the invention achieves near real-time operating file archiving while achieving minimal missed alterations to an operating file.

A further advantage of the invention in its preferred embodiment, is that by intelligently managing the migration of operating files from the input buffer **20** through the output buffer **25** to the storage device **35**, the invention achieves protection of operating files even when the desired storage device is permanently or temporarily unavailable.

INDUSTRIAL APPLICABILITY

The present invention is suited for any application that requires or benefits from near real time file capture, that seeks improved file integrity and/or that seeks efficient management of file storage. For example, the present invention is particularly useful in backup systems, audit trail systems, computer security systems, systems for monitoring computer users and others.

Although the present invention has been described in terms of particular preferred embodiments, it is not limited to those embodiments. Alternative embodiments, examples, and modifications which would still be encompassed by the invention may be made by those skilled in the art, particularly in light of the foregoing teachings.

We claim:

1. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file;

creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

searching the first temporary storage location for the archive file responsive to the occurrence of a first event; and

moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location,

after storing the archive file in the first temporary storage location, updating a database to indicate that the archive file is located in the first temporary storage location;

determining a final destination for the archive file;

moving the archive file from the first temporary storage location to an intermediate storage location;

updating the database to indicate that the archive file is located in the intermediate storage location; and

after moving the archive file to the second storage location, updating the database to indicate that the archive file is located in the second storage location.

2. The method of claim **1** wherein the second storage location includes a personal attached storage device.

3. The method of claim 1 wherein the second storage location includes a network attached storage device.

4. The method of claim 1 wherein the second storage location includes a peer-to-peer storage device.

5. The method of claim 1 wherein the second storage location includes an Internet storage area network.

6. An article of manufacture comprising a non-transitory computer usable medium having computer readable program code for performing the method of claim 1.

7. An article of manufacture comprising a processor configured to perform the method of claim 1.

8. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file;

creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

searching the first temporary storage location for the archive file responsive to the occurrence of a first event; and

moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location,

wherein the first event includes a message from a timer.

9. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file;

creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

searching the first temporary storage location for the archive file responsive to the occurrence of a first event; and

moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location.

wherein the second event includes a message from a timer.

10. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file;

creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

searching the first temporary storage location for the archive file responsive to the occurrence of a first event; and

moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location,

wherein the second event includes a message indicating when the second storage location is available.

11. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file;

creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

searching the first temporary storage location for the archive file responsive to the occurrence of a first event; and

moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location,

wherein said first event is different from said second event.

* * * * *

EXHIBIT B

US009218348B2

(12) **United States Patent**
Roach et al.

(10) **Patent No.: US 9,218,348 B2**
(45) **Date of Patent: *Dec. 22, 2015**

(54) **AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS**

(71) Applicant: **INTEGRITY PC INNOVATIONS, INC.**, Colorado Springs, CO (US)

(72) Inventors: **Warren Roach**, Colorado Springs, CO (US); **Steven R. Williams**, Colorado Springs, CO (US); **Troy J. Reiber**, Colorado Springs, CO (US); **Steven C. Burdine**, Greenwood Village, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/925,768**

(22) Filed: **Jun. 24, 2013**

(65) **Prior Publication Data**

US 2013/0282665 A1 Oct. 24, 2013

Related U.S. Application Data

(63) Continuation of application No. 09/957,459, filed on Sep. 21, 2001, now Pat. No. 8,473,478.

(60) Provisional application No. 60/234,221, filed on Sep. 21, 2000.

(51) **Int. Cl.**
G06F 17/30 (2006.01)
G06F 11/14 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 17/30073*** (2013.01); ***G06F 11/1461*** (2013.01); ***G06F 17/30067*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 17/30073
USPC 707/672
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,959,774	A	9/1990	Davis	
5,086,502	A	2/1992	Malcolm	
5,212,784	A	5/1993	Sparks	
5,241,670	A	8/1993	Eastridge et al.	
5,276,867	A	1/1994	Kenley et al.	
5,479,654	A	12/1995	Squibb	
5,513,112	A *	4/1996	Herring et al.	705/404
5,524,190	A	6/1996	Schaeffer et al.	
5,535,381	A	7/1996	Kopper	
5,604,862	A	2/1997	Midgely et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

EP 259912 B1 3/1988

OTHER PUBLICATIONS

PDC Budtool Live(TM) Available on Solaris; Solaris Port Will Bring Safe, Live Backup Capability to Leading Operating System, PR Newswire, Jun. 6, 1995, 2p.

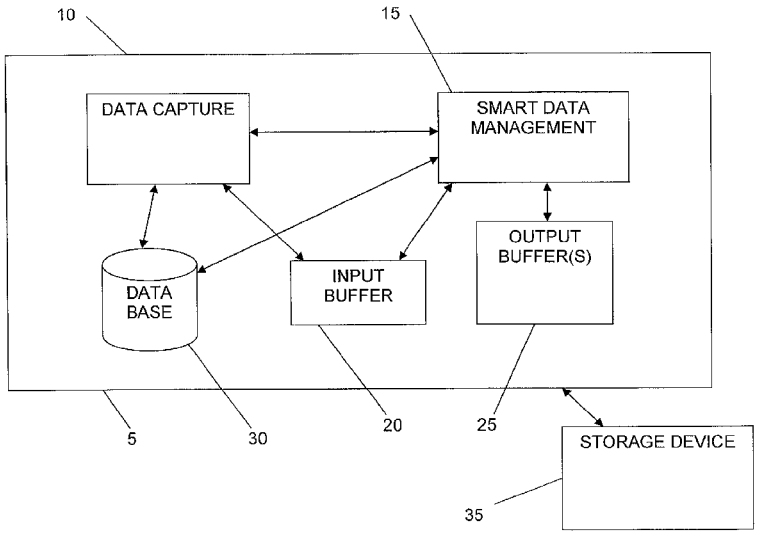
(Continued)

Primary Examiner — Baoquoc To
(74) *Attorney, Agent, or Firm* — Aspire IP; Scott J. Hawranek

(57) **ABSTRACT**

A method for archiving files includes determining when a change in an operating file is imminent, capturing the operating file immediately before the change in the operating file occurs, if the operating file has not already been captured; and capturing the operating file immediately after the change in the operating file has occurred.

31 Claims, 6 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

5,608,865	A		3/1997	Midgely et al.
5,633,999	A		5/1997	Clowes et al.
5,634,052	A		5/1997	Morris
5,638,059	A		6/1997	Pilkington
5,638,509	A	*	6/1997	Dunphy et al. 714/20
5,649,196	A		7/1997	Woodhill et al.
5,668,991	A		9/1997	Dunn et al.
5,721,916	A		2/1998	Pardikar
5,751,997	A		5/1998	Kullick et al.
5,765,173	A		6/1998	Cane et al.
5,771,354	A		6/1998	Crawford
5,813,017	A		9/1998	Morris
5,978,815	A		11/1999	Cabrera et al.
6,014,676	A		1/2000	McClain
6,023,710	A		2/2000	Steiner et al.
6,047,294	A		4/2000	Deshayes et al.
6,101,507	A		8/2000	Cane et al.
6,134,660	A		10/2000	Boneh et al.
6,173,377	B1		1/2001	Yanai et al.
6,212,512	B1		4/2001	Barney et al.
6,269,431	B1		7/2001	Dunham
6,298,319	B1		10/2001	Heile et al.
6,317,845	B1		11/2001	Meyer et al.
6,332,200	B1		12/2001	Meth et al.
6,351,776	B1		2/2002	O'Brien et al.
6,353,878	B1		3/2002	Dunham
6,366,987	B1		4/2002	Tzelnic et al.
6,366,988	B1		4/2002	Skiba et al.
6,434,681	B1		8/2002	Armangau
6,460,055	B1		10/2002	Midgley et al.
6,496,944	B1		12/2002	Hsiao et al.
6,526,418	B1		2/2003	Midgley et al.
6,535,894	B1		3/2003	Schmidt et al.
6,549,992	B1		4/2003	Armangau et al.
6,564,215	B1		5/2003	Hsiao et al.
6,571,280	B1		5/2003	Hubacher
6,611,850	B1		8/2003	Shen
6,615,225	B1		9/2003	Cannon et al.
6,625,623	B1		9/2003	Midgley et al.
6,629,109	B1	*	9/2003	Koshisaka 1/1
6,779,003	B1	*	8/2004	Midgley et al. 1/1
6,802,025	B1		10/2004	Thomas et al.
6,804,689	B1		10/2004	Havrda et al.
6,847,984	B1		1/2005	Midgley et al.
6,983,227	B1	*	1/2006	Thalhammer-Reyero 703/2
7,031,904	B1		4/2006	Wilson et al.
7,117,371	B1		10/2006	Parthasarathy et al.
8,473,478	B2		6/2013	Roach et al.
2002/0107877	A1		8/2002	Whiting et al.

OTHER PUBLICATIONS

Da Silva et al., Performance of a Parallel Network Backup Manager, USENIX Summer 1992 Technical Conference, San Antonio, Texas, USA, Jun. 8-Jun. 12, 1992, pp. 217-225.

Da Silva et al., The Amanda Network Backup Manager, USENIX Seventh Large Installation Systems Administration Conf. (LISA '93), Monterey, California, USA, Nov. 1-Nov. 5, 1993, pp. 170-182.

EpochBackup 7 Administration and User Guide, Mar. 1997, 138p.

Proceedings of Sixth Goddard Conference on Mass Storage Systems and Technologies in cooperation with the Fifteenth IEEE Symposium on Mass Storage Systems, College Park, Maryland, USA, 3/23-3/26/2998, 450p.

Software does live backup, Government Computer News, 13:14, Jul. 11, 1994, p. 48.

Introducing Filo(TM) and Sync, Wayback Machine, Feb. 2000, 5p.

Kolstad, R., A Next Step in Backup and Restore Technology, USENIX Fifth Large Installation Systems Administration Conf. (LISA V), San Diego, California, USA, Sep. 30-Oct. 3, 1991, pp. 73-80.

Norton Ghost(TM) Personal Edition User's Guide, 80p.

Shumway, S., Issues in On-line Backup, USENIX Fifth Large Installation Systems Administration Conf. (LISA V), San Diego, California, USA, Sep. 30-Oct. 3, 1991, pp. 81-88.

Improved SnapBack Live Provides Enhanced Remote-Site Administration and Faster Backup, Business Wire, Feb. 10, 1998, 2p.

Templeman, P., Network Backup and Archival Strategies, AUUGN, 14:5, Oct. 1993, pp. 66-76.

Clapperton, G., Understanding Online Backup, PC Network Advisor, 121:15-18, Aug. 2000.

Van Meter et al., VISA: Netstation's Virtual Internet SCSI Adapter, Jul. 15, 1997, 8p.

Zwicky, E., Further Torture: More Testing of Backup and Archive Programs, USENIX 17th Large Installation Systems Administration (LISA '03), San Diego, California, USA, Oct. 26-Oct. 31, 2003, pp. 7-14.

Tichy, Walter F., "RCS: A System for Version Control" (1984). Computer Science Technical Reports. Paper 394. http://docs.lib.purdue.edu/cstech/394.

* cited by examiner

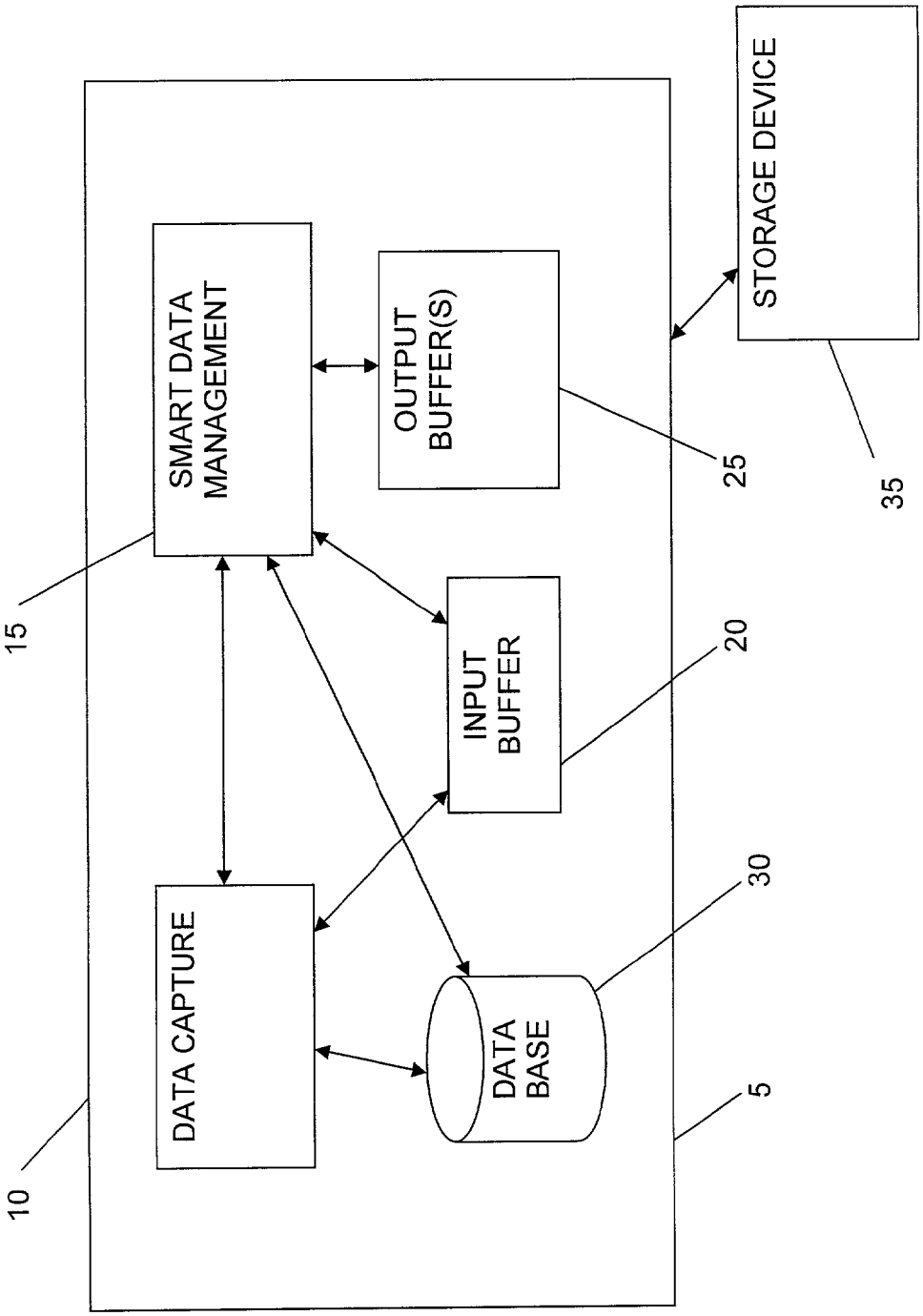

STORAGE DEVICE
SMART DATA MANAGEMENT
OUTPUT BUFFER(S)
INPUT BUFFER
DATA CAPTURE
DATA BASE
10
15
20
25
30
35
5

FIGURE 1

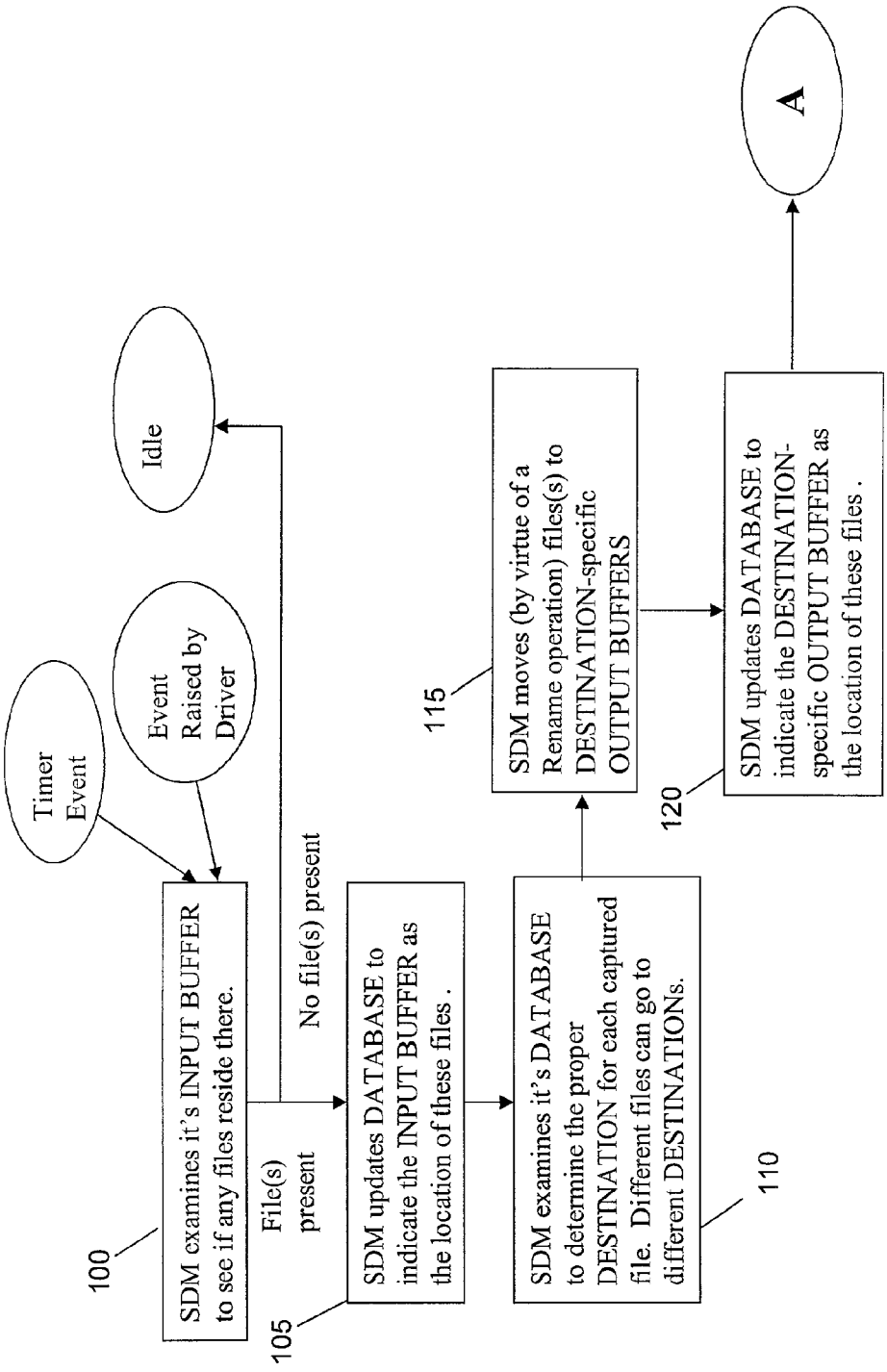

Timer Event
Event Raised by Driver
Idle
100
SDM examines it's INPUT BUFFER to see if any files reside there.
File(s) present
No file(s) present
105
SDM updates DATABASE to indicate the INPUT BUFFER as the location of these files .
SDM examines it's DATABASE to determine the proper DESTINATION for each captured file. Different files can go to different DESTINATIONs.
110
115
SDM moves (by virtue of a Rename operation) files(s) to DESTINATION-specific OUTPUT BUFFERS
120
SDM updates DATABASE to indicate the DESTINATION-specific OUTPUT BUFFER as the location of these files .
A

FIGURE 2A

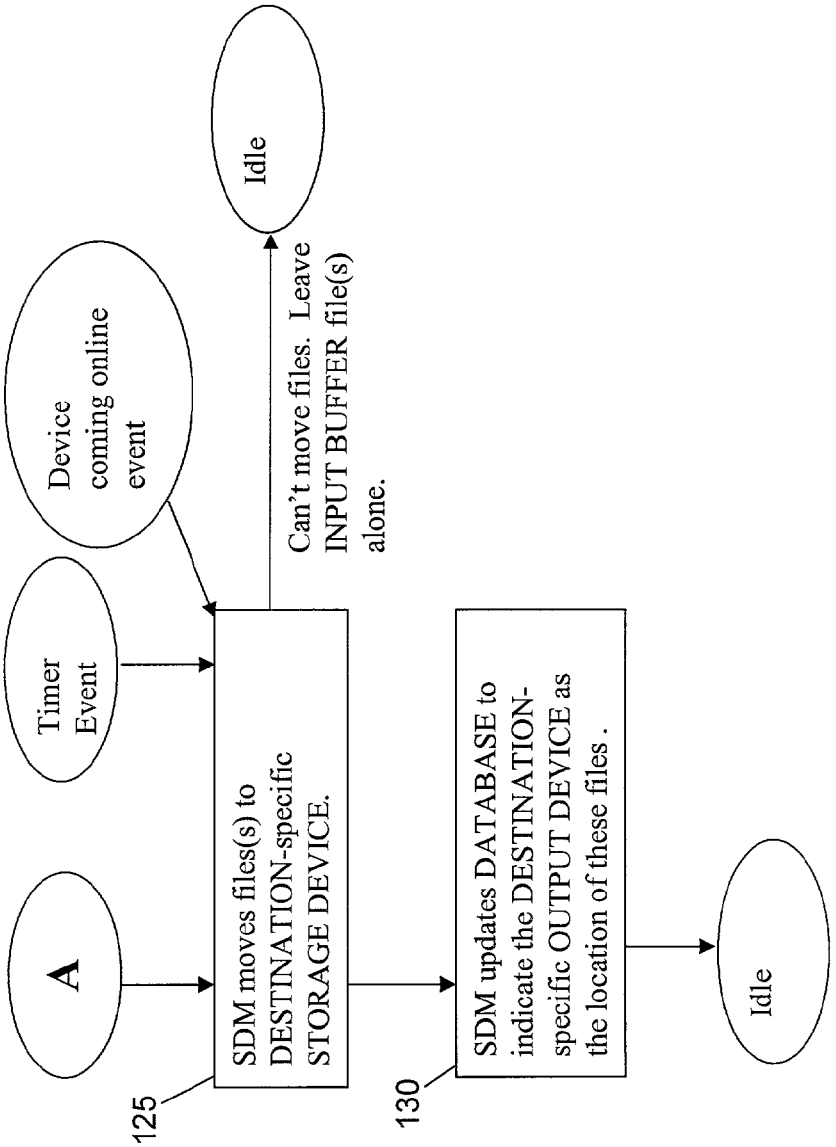

A
Timer Event
Device coming online event
SDM moves files(s) to DESTINATION-specific STORAGE DEVICE.
125
Can't move files. Leave INPUT BUFFER file(s) alone.
Idle
SDM updates DATABASE to indicate the DESTINATION-specific OUTPUT DEVICE as the location of these files .
130
Idle

FIGURE 2B

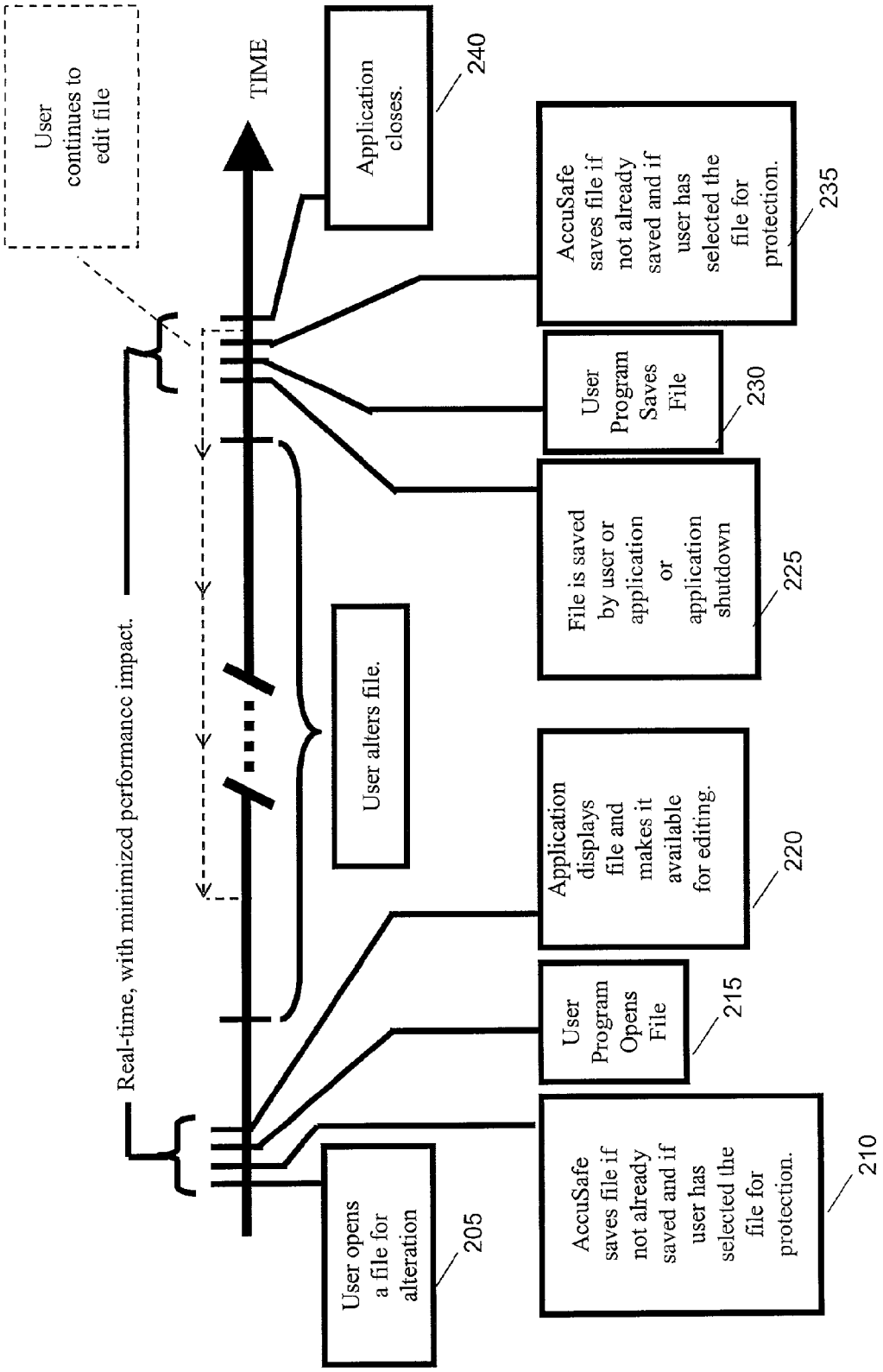

User continues to edit file
TIME
Application closes.
240
AccuSafe saves file if not already saved and if user has selected the file for protection.
235
User Program Saves File
230
File is saved by user or application or application shutdown
225
User alters file.
Application displays file and makes it available for editing.
220
User Program Opens File
215
AccuSafe saves file if not already saved and if user has selected the file for protection.
210
User opens a file for alteration
205
Real-time, with minimized performance impact.

FIGURE 3

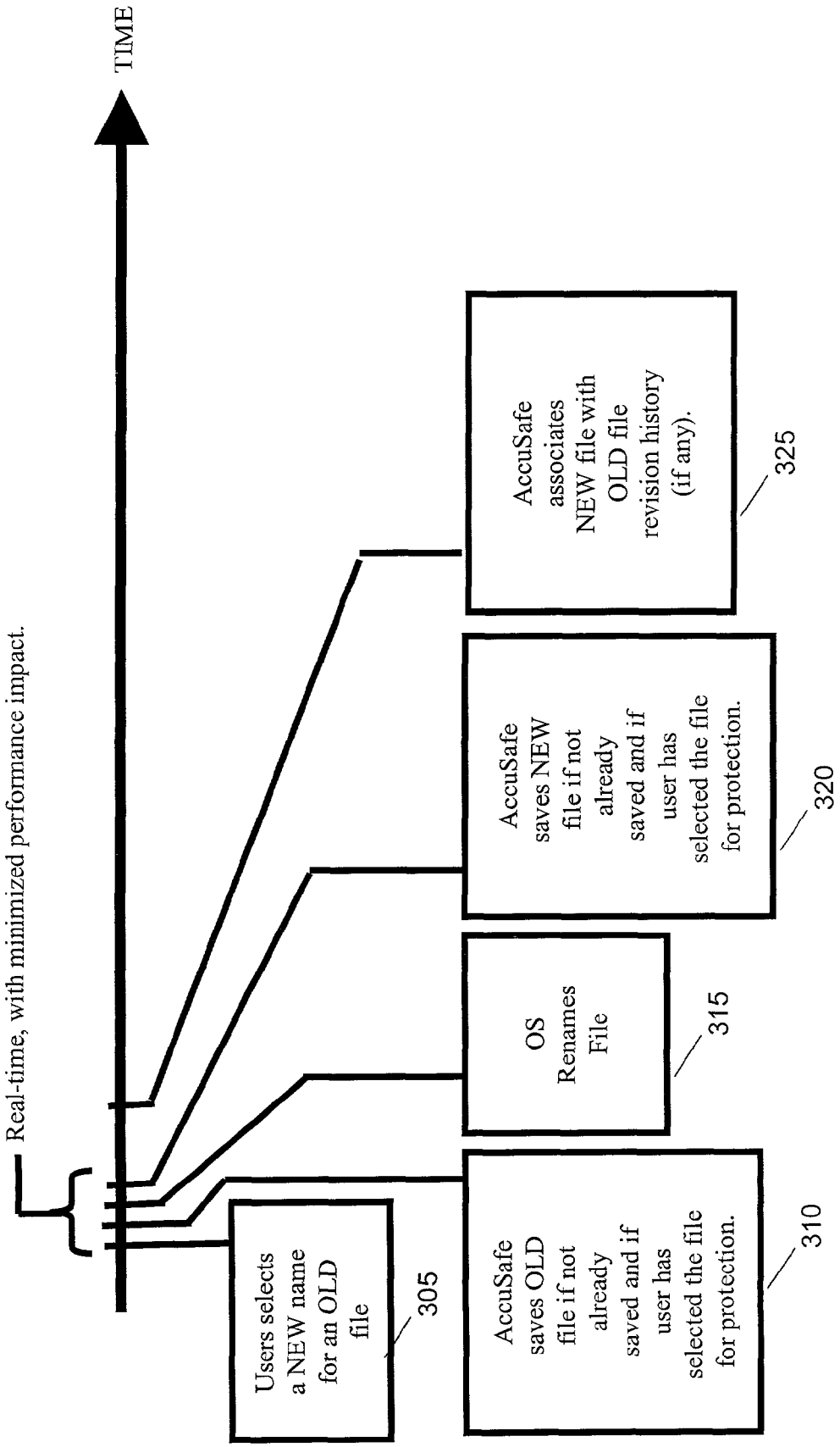

TIME
Real-time, with minimized performance impact.
Users selects a NEW name for an OLD file
305
AccuSafe saves OLD file if not already saved and if user has selected the file for protection.
310
OS Renames File
315
AccuSafe saves NEW file if not already saved and if user has selected the file for protection.
320
AccuSafe associates NEW file with OLD file revision history (if any).
325

FIGURE 4

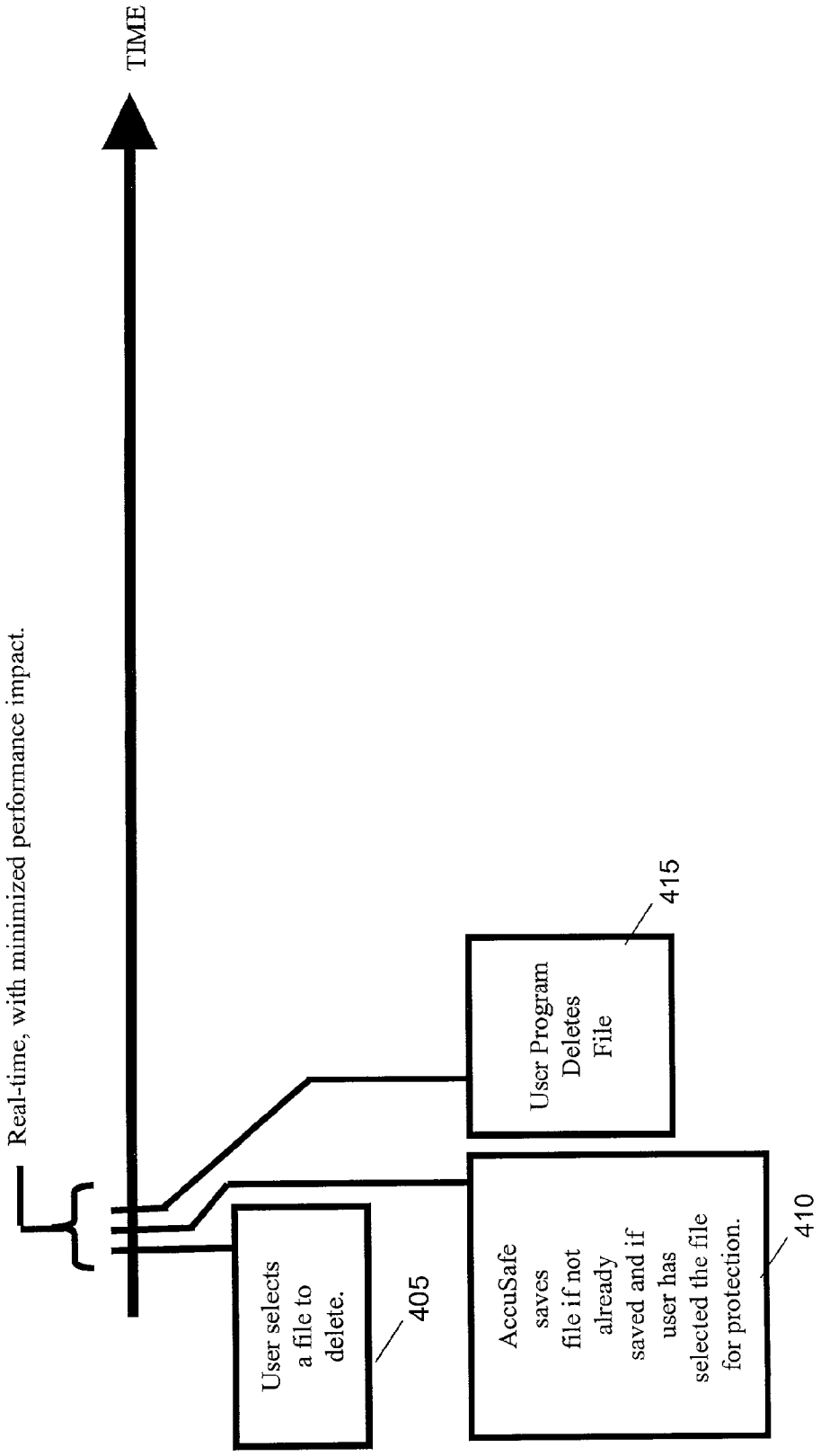

TIME
Real-time, with minimized performance impact.
User selects a file to delete.
405
AccuSafe saves file if not already saved and if user has selected the file for protection.
410
User Program Deletes File
415

FIGURE 5

AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS

This application is a continuation of U.S. patent application Ser. No. 09/957,459 filed Sep. 21, 2001, which claims the benefit of U.S. Provisional Application No. 60/234,221 filed Sep. 21, 2000, each of which is herein incorporated by reference.

FIELD OF THE INVENTION

This invention relates to file preservation, file capture, file management and file integrity techniques for computing devices.

BACKGROUND OF THE INVENTION

One of the greatest challenges faced by information technology (IT) professionals and computer users today, particularly in the business environment is the protection and management of data. Data may be stored on user workstations, e.g., laptop computers, home or office desktop computers, network servers or other devices external to the workstations. Important data may even be stored on hand-held computing devices such as PDAs, PALs and other like devices. Complicating the problem is the fact that the criticality of data is increasing and the difficulty of managing it, protecting it from loss and keeping it available is increasing. This is due to a variety of factors, including: 1) the explosion in data volume, particularly that stored on desktop and laptop computers, 2) the increasing complexity of desktop and laptop computer software and hardware and increasing trends toward a paperless environment were absolute reliance (because paper copies are becoming less the norm) on data integrity is increasingly significant.

Many home computer users do not realize the vulnerability of their computer data. Many that do understand the very real potential for data loss, purchase backup systems whose operation and user interface is often confusing and/or time-consuming to use, dramatically decreasing their effectiveness or dependability. As a result, many computer users remain very much at risk of data loss resulting from hardware and/or software failures, fires, stolen equipment, etc. While these risks are significant, the most frequent cause of data loss is user error (accidental file deletes, file overwrites, errant programs, etc.), to which users remain very vulnerable even with most present day backup systems.

The financial impact of information loss is substantial. As reported by the Safeware Insurance Agency, in 1999 alone, insurance claims for damaged, lost and stolen computers (primarily notebook computers) totaled more than $1.9 billion. This figure does not include the untold billions lost in intellectual capital and time. It is costly to recreate lost data and there are significant related costs such as lost productivity and lost opportunity. Consider, for example, the financial and health related impact of a doctor losing all patient contact information and medical histories due to a hard disk crash or some other type of computer failure. In addition, it is costly to keep desktop and laptop computers up and running in the wake of their increasing complexity.

A variety of products have been developed to address data preservation and integrity issues. These products may be loosely grouped into three categories, manual backup systems, schedule based backup systems and mirroring backup systems.

The least efficient and probably one of the most frequently used backup systems is the manual backup. At times determined by the user, the user selects files to be backed up and either utilizes the built in backup procedure for the corresponding application or manually copies the selected files to a desired backup storage media.

The problems with this method of preserving data are self-evident. Backup procedures are often confusing and may differ from application to application. Accordingly, the user must familiarize itself with the various methods for performing backups. In addition, users may forget to backup or elect not to on a given occasion due to time constraints or other reasons. Manual backups often do not allow the user to continue to use the system during the backup procedure. Furthermore, data stored to the backup media is really only a "snapshot" of the data at the time that the backup is performed. Any changes made between manual backups would be lost if there was a failure on the computer's storage device.

Schedule based backup systems typically perform backups according to a schedule either set by the user or preset by the backup software. One of the major disadvantages of each type of schedule-based backup system is that as with manual backups, they miss work done between schedule points. This may cause the user to loose critical information as they work between schedule points. Another disadvantage of schedule-based backups systems is that they are frequently confusing and cumbersome for the user. Still another disadvantage of schedule-based backup systems is that they function poorly if at all when the backup storage device is unavailable, i.e., they cannot be written to due to a communications error or because the device has reached its capacity, is bandwidth limited, or is non-operational for some other reason.

Mirroring is a technique typically applied to disk based backup systems. Mirroring backup systems are the most comprehensive in that everything that happens to the source storage device immediately happens to the backup storage device. That is the backup drive becomes a mirror image of the source drive. Accordingly, if a failure occurs on the source disk, processing can be switched to the backup disk with little or no service interruption.

The strongest advantage of mirroring systems is also their strongest disadvantage. Because there is no operational discrimination, if a file is accidentally deleted from the source disk, it is deleted and cannot be preserved on the backup disk. Likewise, if a virus infects the source disk it is likely to infect the backup disk. Another disadvantage of mirroring systems is that separate backup disks are required for each source disk, doubling the disk requirement for the system. The backup disk must be at least as large as the source disk and the disks must be configured with identical volume mapping. Any extra space that may be present on the backup disk is unavailable.

All of these methods require that the user specify which files/directories to back up, but many users have no concept of files and directories in their thought process, much less are they able to correlate a particular application (e.g. Microsoft Excel) with the kinds and locations of files they generate. These systems simply require too much user knowledge, and too much user intervention. The backup user's risk increases dramatically the lower his computer knowledge may be.

In view of the foregoing, there is a need for a file capture, preservation and management system that captures files just before and/or just after they have been changed to minimize loss of data between backup events. There is also a need for file capture and preservation system that captures files even when the destination storage medium for the files is unavailable. There is a further need for a system that allows users to recover easily and quickly from any type of information loss, including simple user errors, failed software installations or updates, hardware failures (attached storage devices), and

lost or stolen laptop computers. Users should be able to recover on their own, without the intervention of the IT staff, and their backup systems should be as "behind the scenes" as possible, requiring little user attention and extremely small amounts of user computer knowledge.

SUMMARY OF THE INVENTION

It is an object of the invention to a file capture, preservation and management method and apparatus that captures files just before and/or just after the files are changed.

It is another object of the invention to provide a file capture, preservation and management method and apparatus that has an imperceptible impact on system performance from the user's point of view.

It is a further object of the invention to provide a file capture, preservation and management method and apparatus that captures and stores files even when there is no connection to the desired storage location.

Still another object of the invention is to provide a file capture, preservation and management method and apparatus that captures and stores files even when the desired storage location is unavailable.

In accordance with an aspect of the invention, a method for archiving files is provided. The method includes, in a computing device, detecting an instruction from a resident program to perform an operation on an operating file. Upon detection of the instruction, capturing the operating file temporally proximate to the operation being performed on the operating file.

In accordance with another aspect of the invention, a method for moving files from a first storage location to a second storage location is provided. The method includes, in a computing device, searching a first storage location for files responsive to the occurrence of a first event and moving the files from the first storage location to the second storage location responsive to a second event.

In accordance with still another aspect of the invention, a method for archiving files is provided. The method includes detecting an instruction from a resident program to perform an operation on an operating file. The method further includes creating an archive file from the operating file and storing the archive file in a first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction. In keeping with the method, the first storage location is searched for an archive file responsive to the occurrence of a first event. The archive file is then moved from the first storage location to the second storage location responsive to a second event.

The accompanying figures show illustrative embodiments of the invention from which these and other of the objectives, novel features and advantages will be readily apparent.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a block diagram of a computing device in accordance with the present invention.

FIG. **2**A is a flow chart depicting a process for moving files in accordance with the present invention.

FIG. **2**B is a flow chart showing another process for moving files in accordance with the invention.

FIG. **3** is a time line illustrating a sequence of events in an exemplary operation in accordance with the invention.

FIG. **4** is a time line illustrating a sequence of events in another exemplary operation in accordance with the invention.

FIG. **5** is a time line illustrating a sequence of events in still another exemplary operation in accordance with the invention.

DETAILED DESCRIPTION OF THE EMBODIMENT

Definitions

Operating System (OS)—A computer program that allocates system resources such as memory, disk space, and processor usage and makes it possible for the computer to boot up to a human user interface allowing the user to interact with the computer and control its operation.

Operating File—a system or user file.

Archive File—a file containing all of the data of an operating file in a native or altered format and/or a file containing at least some of the data of an operating file and including references to the location of the remainder of the data of the operating file.

Computing Device—a personal computer, a laptop or notebook computer, a server, a hand-held computing device, a PDA or a PAL. The term computing device is not specific to the kind of operating system being run on such computing device, and includes devices running Microsoft operating systems, Apple Macintosh operating systems, UNIX operating systems, Linux operating systems, and other operating systems.

Storage Location—any storage device, or a buffer, folder, directory or designated area on a storage device.

Personal Attached Storage Device—any internal or external storage device connected to a computing device.

Network Attached Storage Device—any storage device connected directly to a network to which a first computing device is also temporarily or permanently connected, or any storage device connected to a second computing device that is also temporarily or permanently connected to the network to which the first computing device is temporarily or permanently connected.

Internet storage area network—any storage area (device, collection of devices, etc.) that can be accessed by the computing device when the computing device is temporarily or permanently connected to the Internet.

Peer-to-Peer Storage Device—any storage area (device, collection of devices, etc.) that can be accessed by the computing device when it is sharing resources with other network or Internet accessible computers.

Resident Program—an operating system (OS) or other program that has control over file operations such as "read", "write", "save", "rename", "delete", "copy", "move", "open", "close", etc.

User Program—an application software program or other computing program installed by the user or by the computer manufacturer for user creation of desired data, documents, or other information that is designed to enhance the functionality and/or enjoyment and/or usability of the computing device. The present invention is directed to an apparatus and/or method for file capture, preservation and management. The invention includes a file capture aspect and smart data management aspect. The invention may be realized as a method and/or an apparatus. More particularly, the invention may be realized as a set of program code instructions stored on a computer usable medium, a set of program code instructions embodied in a signal for transmitting computer information, and a processor and/or computing device configured as described herein.

FIG. **1** depicts a block diagram in accordance with the present invention comprising a computing device **5** including a file capture block **10** (or file capturer), a smart data management block **15** (or smart data manager), an input buffer **20**, output buffer(s) **25**, and a database **30**. A storage device **35** is also provided and may be either internal or external to computing device **5**. The invention functions in conjunction with a resident program on computing device **5**.

In accordance with an embodiment of the invention, file capture block **10** detects an instruction to perform an operation on an operating file initiated by the resident program of computing device **5**. At a moment temporally proximate to when the resident program actually performs the operation, i.e., just before and/or just after the operation is performed on the operating file, or, more preferably, the instant before and/or the instant after the operating file is changed, file capture block **10** captures the operating file or portions thereof. Preferably, the operating file is captured within a few clock cycles of the detection of the instruction.

In keeping with a preferred aspect of the invention, file capture block **10** causes the location of the captured operating file to be recorded in database **30**. The continued process of recording information about captured operating files, or portions thereof, in database **30** creates a record of each version of the operating file, which may be accessed by the user or by other programs.

File capture is preferably executed by creating an archive file from the operating file. The archive file is preferably stored in a temporary storage location, internal or external to the computer, such as input buffer **20**. However, the archive file may be stored directly in storage device **35**. In accordance with a preferred aspect of the invention, storage device **35** may be a personal attached storage device, a network attached storage device, an Internet storage area network, a peer-to-peer storage device, or other storage device.

In keeping with a preferred aspect of the invention, smart data management block **15** manages the migration of the archive file from the input buffer **20** through the output buffers **25** to storage device **35**. This migration may take place either synchronously or asynchronously with the file capture procedures described herein. The time duration from a file arriving in input buffer **20** and when it arrives on archive storage device **35** is managed by the smart data management block **15**. More particularly smart data management block **15** regularly examines input buffer **20** for the presence of archive files. Smart data management block **15** performs this examination upon the occurrence of an event, e.g., messages from the file capture block **10** and/or various messages from the resident program(s), messages from an input buffer timer sent at time intervals controlled by a timer or at time intervals selected by the user. Optionally, smart data management block **15** may then examine database **30** to determine a defined storage location for each of the archive files stored in input buffer **20**. Each archive file stored in the input buffer **20** may be directed to the same storage location or to different storage locations and archive files may be directed to multiple storage locations for redundancy. Preferably, smart data management block **15** moves the archive files to one or more output buffers **25**. More preferably each archive file is moved to output buffer(s) **25** corresponding to the final storage location(s) for that archive file. Alternatively, all archive files may be moved to a single common output buffer **25** if desired. Upon the occurrence of an event, and/or at defined time intervals, smart data management block **15** moves the archive files from the output buffers **25** to their respective storage device(s) **35**. Exemplary events include but are not limited to messages indicating when storage device **35** is connected and ready for use, messages indicating when storage device **35** is inserted/removed, full, defective, etc., and messages indicating when storage device **35** is disconnected or unavailable, and messages from a storage device timer sent at time intervals controlled by the timer or at time intervals controlled by the user. The input buffer timer and the storage device timer may operate synchronously or non-synchronously.

Under certain conditions, smart data management block **15** may be unable, or may elect not to move the archive files. For example, if storage device **35** is unavailable then smart data management block **15** will not move the archive files to storage device **35**. Among the conditions that may cause storage device **35** to be unavailable are i) storage device **35** is disconnected from computing device **5**, ii) the connection between storage device **35** and computing device **5** is faulty or unacceptably slow, iii) storage device **35** is full, or iv) storage device **35** is malfunctioning. In addition, smart data management block **15** may also regulate movement of archive files according to time schedules set by the user, by monitoring connection bandwidth availability and moving files only during times of high bandwidth availability, or by monitoring other factors including messages that may received from storage location server requests for archive file transmittal.

A preferred operational mode for smart data management block **15** is illustrated in the flowcharts of FIGS. **2**A and **2**B. In step **100** of FIG. **2**A, smart data manager **15** examines input buffer **20** to determine whether any archive files are stored therein. If no archive files are present, smart data manager **15** rests idle until the next event occurs. If archive files are detected, in step **105**, smart data manager **15** updates database **25** to indicate the location of the archive files; that is, to indicate that the archive files are resident in input buffer **20**. In step **110**, smart data manager **15** examines database **30** to determine the proper destination for each archive file. In step **115**, smart data manager **15** moves the archive files to output buffers **25**. In step **120**, smart data manager **15** updates database **30** to indicate that the archive files are now stored in the output buffer.

In FIG. **2**B in step **125**, the archive files are moved to one or more storage devices **30**. If smart data manager **15** is unable to move the archive files to any of the storage devices **30**, smart data manager **15** rests idle and does not move the archive files until it is notified that the storage device is available. Accordingly, the archive files remain in either input buffer **20** or output buffer **25** until smart data management block **15** is notified. In step **130** smart data manager **15** updates database **25** to indicate that the archive files are stored in one or more storage devices **30**.

Use Specific to User Program Operations

The following examples are directed to embodiments of the invention specific to operations performed by a user program. The file capture, preservation and management processes of the invention are not limited to execution with the exemplary operation discussed below. The processes of the invention are preferably executed when a resident program causes a change or a change to be imminent in the operating file. Therefore, the following examples are intended to be exemplary only and non-limiting.

File Capture at File Open

As illustrated in FIG. **3**, in step **205**, the user or a program selects an "open" operation to open an operating file and an instruction to perform that "open" operation on the operating file is sent to the resident program. In step **210**, file capture block **10** detects the instruction and captures the operating file. Optionally, prior to capturing the operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine

whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** creates an archive file and stores the archive file in a storage location such as input buffer **20** or storage device **35** just before the resident program opens the operating file. Preferably, file capture block **10** stores the archive file in input buffer **20**. In step **215** the resident program opens the operating file and in step **220** the user program displays the operating file as originally requested, e.g. Microsoft Word, and makes it available for the user to alter, e.g., edit a word processing document, amend or add to a database, etc. Step **210** is performed by momentarily delaying the execution of step **215** in such a manner as to have little or no perceptible impact on system performance from the user's point of view.

In step **225**, the user program begins a process to save the altered operating file and an instruction to save the altered operating file is sent to the resident program. In step **230** the resident program saves the altered operating file pursuant to the instruction. In step **235**, immediately after the altered operating file is saved by the resident program, file capture block **10** captures the altered operating file, preferably by creating and storing an archive file of the altered operating file in input buffer **20**. In accordance with a preferred feature of the invention, file capture block **10** may save the archive file in such a way that previous revisions of the operating file are retained. That is, every time the operating file is changed, file capture block **10** saves an archive file and database **30** is updated with information about the archive file. Accordingly, over time, a plurality of archive files may be created from the original operating file. Each archive file represents a revision of the original operating file.

File Capture in the "RENAME" Operation

As illustrated in FIG. **4**, step **305**, in performing an operating file rename operation, the user or a program generates an instruction for the resident program to select a new name for an old operating file. In step **310**, file capture block **10** detects the instruction and captures the old operating file. Optionally, prior to capturing the old operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** creates an archive file of the old operating file and stores the archive file in a storage location such as storage device **35** or, more preferably, input buffer **20** just before the resident program renames the old operating file. In step **315** the resident program renames the old operating file, thus creating a new operating file. Immediately after the old operating file is renamed, file capture block **10** captures the new operating file. Optionally, prior to capturing the new operating file, file capture block **10** may determine whether the new operating file has previously been archived, whether the user has selected the new operating file for protection, or other matching conditions exist. Like the archive file for the old operating file, the archive file for the new operating file is preferably stored in input buffer **20**. In step **325** file capture block **10** and smart data management block **15** associate or link the new operating file with each of the versions of the old operating file to create a continuous operating file revision history.

File Capture in the "Delete" Operation

FIG. **5** illustrates the file capture process in the delete operation. In step **405**, the user or a program identifies an operating file to delete and generates an instruction to the resident program. In step **410**, file capture block **10** detects the instruction and captures the operating file just before it is deleted in step **415**. Optionally, prior to capturing the operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** preferably captures the operating file. In step **420**, the resident program deletes the operating file.

As shown by the examples given, a clear advantage of the invention is, regardless of the operation being performed, after each file capture step, file capture block **10** preferably updates database **30** to indicate the location of the corresponding archive file. Database **30** may keep track of multiple versions of an operating file, any of which may be accessed at the request of the user or other program.

Another advantage of the invention is that by capturing the operating file just before and/or just after an operation is performed thereon, the invention achieves near real-time operating file archiving while achieving minimal missed alterations to an operating file.

A further advantage of the invention in its preferred embodiment, is that by intelligently managing the migration of operating files from the input buffer **20** through the output buffer **25** to the storage device **35**, the invention achieves protection of operating files even when the desired storage device is permanently or temporarily unavailable.

INDUSTRIAL APPLICABILITY

The present invention is suited for any application that requires or benefits from near real time file capture, that seeks improved file integrity and/or that seeks efficient management of file storage. For example, the present invention is particularly useful in backup systems, audit trail systems, computer security systems, systems for monitoring computer users and others.

Although the present invention has been described in terms of particular preferred embodiments, it is not limited to those embodiments. Alternative embodiments, examples, and modifications which would still be encompassed by the invention may be made by those skilled in the art, particularly in light of the foregoing teachings.

We claim:

1. A method for archiving files, comprising steps of (a) to (i) following:

(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file;

(b) the step of creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

(c) the step of identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event;

(d) the step of transmitting the archive file to a second storage location responsive to a second event, the second storage location being an intermediate or a permanent storage location;

(e) after storing the archive file in the temporary storage location, updating a database to indicate that the archive file is located in the temporary storage location;

(f) determining a final destination for the archive file;

(g) moving the archive file from the temporary storage location to an intermediate storage location;

(h) updating the database to indicate the archive file is located in the intermediate storage location; and

(i) after moving the archive file to the second storage location, updating the database to indicate the archive file is located in the second storage location.

2. The method of claim **1**, wherein the second storage location includes a personal attached storage device.

3. The method of claim **1**, wherein the second storage location includes a network attached storage device.

4. The method of claim **1**, wherein the second storage location includes a peer-to-peer storage device.

5. The method of claim **1**, wherein the second storage location includes an internet storage area network.

6. An article of manufacture comprising a computer usable medium having computer readable program code for performing the method of claim **1**.

7. An article of manufacture comprising a processor configured to perform the method of claim **1**.

8. A method for archiving files, comprising the steps of:

(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file;

(b) the step of creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

(c) the step of identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event;

(d) the step of transmitting the archive file to a second storage location responsive to a second event, the second storage location being an intermediate or a permanent storage location;

(e) after storing the archive file in the temporary storage location, updating a database to indicate that the archive file is located in the temporary storage location;

(f) the step of moving the archive file from the temporary storage location to an output buffer;

(g) the step of updating the database to indicate that the archive file is located in the output buffer;

(h) the step of transmitting the archive file from the output buffer to the intermediate or the permanent storage location; and

(i) the step of updating the database to indicate that the archive file is located in the intermediate or the permanent storage location.

9. The method of claim **8**, wherein the second storage location includes a personal attached storage device.

10. The method of claim **8**, wherein the second storage location includes a network attached storage device.

11. The method of claim **8**, wherein the second storage location includes an Internet storage area network.

12. A method for archiving files, comprising steps of (a) to (d) following:

(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file;

(b) the step of creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

(c) the step of identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event; and

(d) the step of transmitting the archive file to a second storage location responsive to a second event, the second storage location being an intermediate or a permanent storage location, wherein the second event includes a message indicating when the second storage location is available.

13. A method for archiving files, comprising steps of (a) to (d) following:

(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file;

(b) the step of creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

(c) the step of identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event; and

(d) the step of transmitting the archive file to a second storage location responsive to a second event, the second storage location being an intermediate or a permanent storage location, wherein the second event includes a message from a timer.

14. A method for archiving files, comprising steps of (a) to (d) following:

(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file;

(b) the step of creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

(c) the step of identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event; and

(d) the step of transmitting the archive file to a second storage location responsive to a second event, the second storage location being an intermediate or a permanent storage location, wherein the first event includes a message from a timer.

15. A method for archiving files, comprising steps of (a) to (d) following:

(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file;

(b) the step of creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

(c) the step of identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event; and

(d) the step of transmitting the archive file to a second storage location responsive to a second event, the second storage location being an intermediate or a permanent storage location, wherein the first event is different from the second event.

16. The method of claim **15**, wherein the resident program is separate from an operating system.

17. The method of claim **16**, where the resident program comprises a program having control over file operations on one or more operating files.

18. The method of claim **15**, wherein the operation comprises one of read, write, save, rename, delete, copy, move, open, close, or create operations.

19. The method of claim **15**, wherein the operating file comprises one of a system file or a user file.

20. The method of claim **15**, wherein the archive file comprises at least some data of the operating file.

21. The method of claim **20**, wherein the archive file further comprises a reference to a location of a remainder of the data of the operating file.

22. The method of claim **15**, wherein the archive file comprises data of the operating file in one or a combination of a native format and an altered format.

23. The method of claim **15**, wherein the temporary storage location is located in a storage device internal to the computing device.

24. The method of claim **23**, wherein the temporary storage location comprises a buffer located in the storage device.

25. The method of claim **15**, wherein the temporary storage location is located in a storage device external to the computing device.

26. The method of claim **15**, wherein the first event comprises one of the following: a message from a process that created the archive file from the operating file, a message from the resident program, or a message from a timer.

27. The method of claim **15**, wherein the second event comprises one of the following: a message indicating when the intermediate or the permanent storage location is available, a message indicating when the intermediate or the permanent storage location is inserted, a message indicating high-bandwidth availability, a message from a server associated with the intermediate or the permanent storage location, or a message from a timer.

28. The method of claim **15**, wherein the intermediate or the permanent storage location is located in one of a personal attached storage device, a network attached storage device, a Internet storage area network, or a peer-to-peer storage device.

29. A non-transitory computer readable medium comprising program instruction for performing the method of claim **15**.

30. The method of claim **15**, wherein the intermediate or the permanent storage location is accessible by the computing device through the Internet.

31. The method of claim **15**, wherein the second storage location is the permanent storage location.

* * * * *

EXHIBIT C

US010585850B2

(12) **United States Patent**
Roach et al.

(10) **Patent No.: US 10,585,850 B2**
(45) **Date of Patent: Mar. 10, 2020**

(54) **AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS**

(71) Applicant: **DATANET, LLC**, Colorado Springs, CO (US)

(72) Inventors: **Warren Roach**, Colorado Springs, CO (US); **Steven R. Williams**, Colorado Springs, CO (US); **Troy J. Reiber**, Colorado Springs, CO (US); **Steven C. Burdine**, Middleton, WI (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1008 days.

(21) Appl. No.: **14/970,044**

(22) Filed: **Dec. 15, 2015**

(65) **Prior Publication Data**

US 2016/0103843 A1 Apr. 14, 2016

Related U.S. Application Data

(63) Continuation of application No. 13/925,768, filed on Jun. 24, 2013, now Pat. No. 9,218,348, which is a (Continued)

(51) **Int. Cl.**
G06F 17/30 (2006.01)
G06F 16/11 (2019.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06F 16/113*** (2019.01); ***G06F 11/1461*** (2013.01); ***G06F 16/10*** (2019.01); ***G06F 16/86*** (2019.01)

(58) **Field of Classification Search**
CPC G06F 11/1469
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,959,774 A 9/1990 Davis
5,086,502 A 2/1992 Malcolm
(Continued)

FOREIGN PATENT DOCUMENTS

EP 259912 B1 3/1988

OTHER PUBLICATIONS

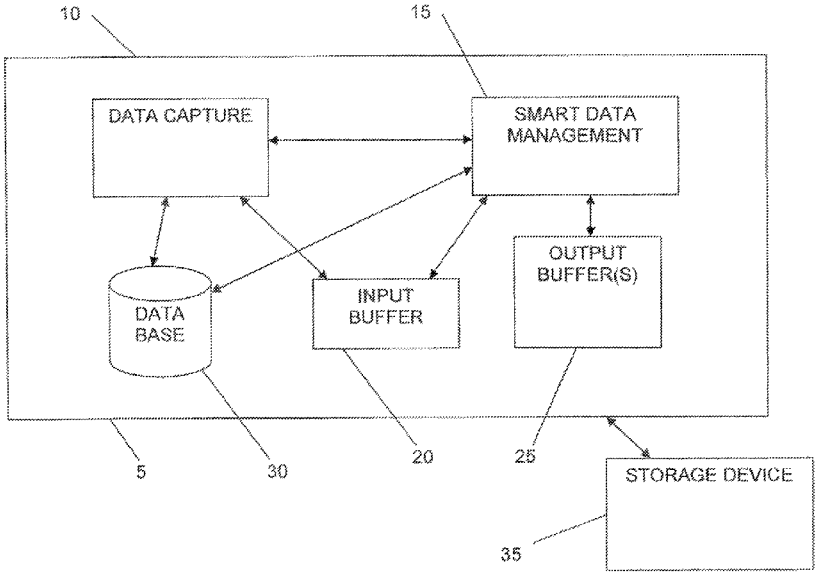
PDC Budtool Live(TM) Available on Solaris; Solaris Port Will Bring Safe, Live Backup Capability to Leading Operating System, PR Newswire, Jun. 6, 1995, 2p.
(Continued)

Primary Examiner — Baoquoc N To
(74) *Attorney, Agent, or Firm* — Aspire IP, LLC

(57) **ABSTRACT**

A method of restoring a file to a previous version of the file, where a current version of the file being available at a local storage location is disclosed. The method includes receiving a request to display a list of captured revisions for a file; displaying a list of captured revisions for said file; receiving a selection from said list indicating a revision of said file to restore; previewing selected revision of said file, wherein said selected revision of said file is received from an Internet storage area network; receiving another selection from said list indicating another revision of said file to restore; previewing selected another revision of said file, wherein said selected another revision of said file is received from a network attached storage location; and restoring selected another revision of said file following the previewing step.

21 Claims, 6 Drawing Sheets



Related U.S. Application Data

continuation of application No. 09/957,459, filed on Sep. 21, 2001, now Pat. No. 8,473,478.

(60) Provisional application No. 60/234,221, filed on Sep. 21, 2000.

(51) **Int. Cl.**

G06F 16/10	(2019.01)
G06F 16/84	(2019.01)
G06F 11/14	(2006.01)

(58) **Field of Classification Search**

USPC 707/671, 655

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,212,784	A		5/1993	Sparks
5,241,670	A		8/1993	Eastridge et al.
5,276,867	A		1/1994	Kenley et al.
5,479,654	A		12/1995	Squibb
5,513,112	A		4/1996	Herring et al.
5,524,190	A		6/1996	Schaeffer et al.
5,535,381	A		7/1996	Kopper
5,604,862	A		2/1997	Midgely et al.
5,608,865	A		3/1997	Midgely et al.
5,634,052	A		3/1997	Morris
5,633,999	A		5/1997	Clowes et al.
5,638,059	A		6/1997	Pilkington
5,638,509	A		6/1997	Dunphy et al.
5,649,196	A		7/1997	Woodhill et al.
5,668,991	A		9/1997	Dunn et al.
5,675,802	A	*	10/1997	Allen G06F 8/71 717/103
5,721,916	A		2/1998	Pardikar
5,751,997	A		5/1998	Kullick et al.
5,765,173	A		6/1998	Cane et al.
5,771,354	A	*	6/1998	Crawford G06F 9/5061 707/999.202
5,774,717	A		6/1998	Porcaro
5,778,395	A	*	7/1998	Whiting G06F 11/1464
5,813,017	A		9/1998	Morris
5,909,700	A		6/1999	Bitner et al.
5,978,815	A		11/1999	Cabrera et al.
5,991,771	A		11/1999	Falls et al.
5,991,772	A		11/1999	Doherty et al.
6,014,676	A		1/2000	McClain
6,023,710	A		2/2000	Steiner et al.
6,032,227	A		2/2000	Shaheen et al.
6,047,294	A		4/2000	Deshayes et al.
6,101,507	A		8/2000	Cane et al.
6,106,570	A		8/2000	Mizuhara
6,112,024	A		8/2000	Almond et al.
6,134,660	A		10/2000	Boneh et al.
6,151,674	A		11/2000	Takatani
6,173,377	B1		1/2001	Yanai et al.
6,212,512	B1		4/2001	Barney et al.
6,269,371	B1		7/2001	Ohnishi
6,269,431	B1		7/2001	Dunham
6,298,319	B1		10/2001	Heile et al.
6,317,845	B1		11/2001	Meyer et al.
6,332,200	B1		12/2001	Meth et al.
6,351,776	B1		2/2002	O'Brien et al.
6,353,878	B1		3/2002	Dunham
6,366,987	B1		4/2002	Tzelnic et al.
6,366,988	B1		4/2002	Skiba et al.
6,434,681	B1		8/2002	Armangau
6,460,055	B1		10/2002	Midgley et al.
6,496,944	B1		12/2002	Hsiao et al.
6,505,200	B1		1/2003	Ims et al.
6,526,418	B1		2/2003	Midgley et al.
6,535,894	B1		3/2003	Schmidt et al.
6,549,992	B1		4/2003	Armangau et al.
6,564,215	B1		5/2003	Hsiao et al.
6,571,280	B1		5/2003	Hubacher
6,611,850	B1		8/2003	Shen
6,615,225	B1		9/2003	Cannon et al.
6,625,623	B1		9/2003	Midgley et al.
6,629,109	B1		9/2003	Koshisaka
6,779,003	B1		8/2004	Midgley et al.
6,802,025	B1		10/2004	Thomas et al.
6,804,689	B1		10/2004	Havrda et al.
6,847,984	B1		1/2005	Midgley et al.
6,983,227	B1		1/2006	Thalhammer-Reyero
7,031,904	B1		4/2006	Wilson et al.
7,117,371	B1		10/2006	Parthasarathy et al.
8,473,478	B2		6/2013	Roach et al.
2002/0107877	A1		8/2002	Whiting et al.

OTHER PUBLICATIONS

Da Silva et al., Performance of a Parallel Network Backup Manager, USENIX Summer 1992 Technical Conference, San Antonio, Texas, USA, Jun. 8-Jun. 12, 1992, pp. 217-225.

Da Silva et al., The Amanda Network Backup Manager, USENIX Seventh Large Installation Systems Administration Conf. (LISA '93), Monterey, California, USA, Nov. 1-Nov. 5, 1993, pp. 170-182.

EpochBackup 7 Administration and User Guide, Mar. 1997, 138p.

Proceedings of Sixth Goddard Conference on Mass Storage Systems and Technologies in cooperation with the Fifteenth IEEE Symposium on Mass Storage Systems, College Park, Maryland, USA, Mar. 23-Mar. 26, 1998, 450p.

Software does live backup, Government Computer News, 13:14, Jul. 11, 1994, p. 48.

Introducing Filo(TM) and Sync, Wayback Machine, Feb. 2000, 5p.

Kolstad, R., A Next Step in Backup and Restore Technology, USENIX Fifth Large Installation Systems Administration Conf. (LISA V), San Diego, California, USA, Sep. 30-Oct. 3, 1991, pp. 73-80.

Norton Ghost(TM) Personal Edition User's Guide, 80p.

Shumway, S., Issues in On-line Backup, USENIX Fifth Large Installation Systems Administration Conf. (LISA V), San Diego, California, USA, Sep. 30-Oct. 3, 1991, pp. 81-88.

Improved SnapBack Live Provides Enhanced Remote-Site Administration and Faster Backup, Business Wire, Feb. 10, 1998, 2p.

Templeman, P., Network Backup and Archival Strategies, AUUGN, 14:5, Oct. 1993, pp. 66-76.

Clapperton, G., Understanding Online Backup, PC Network Advisor, 121:15-18, Aug. 2000.

Van Meter et al., VISA: Netstation's Virtual Internet SCSI Adapter, Jul. 15, 1997, 8p.

Zwicky, E., Further Torture: More Testing of Backup and Archive Programs, USENIX 17th Large Installation Systems Administration (LISA '03), San Diego, California, USA, Oct. 26-Oct. 31, 2003, pp. 7-14.

Tichy, Walter F., "RCS: A System for Version Control," Computer Science Technical Reports, 1984, Paper 394, http://doc.lib.purdue.edu/cstech/394.

Kistler et al., "Disconnected Operation in the Coda File," ACM Trans on Computer Systems, 10:1, Feb. 1992, pp. 3-25.

Kistler, James J., Disconnected Operation in a Distributed File System, May 1993, 270pgs.

Munson et al., "Sync: A Java Framework for Mobile Collaborative Applications," Computer, 30:6, Jun. 1997, pp. 59-66.

Satyanarayanan et al., "Experience with Disconnected Operation in a Mobile Computer Environment," Proceedings of the USENIX Mobile & Location-Independent Computing Symposium, Cambridge, MA, USA, Aug. 2-3, 1993, 21pgs.

Satyanarayanan et al., "Coda: A Highly Available File System for a Distributed Workstation Environment," IEEE Trans on Comuters, 39:4, Apr. 1990, pp. 447-459.

* cited by examiner

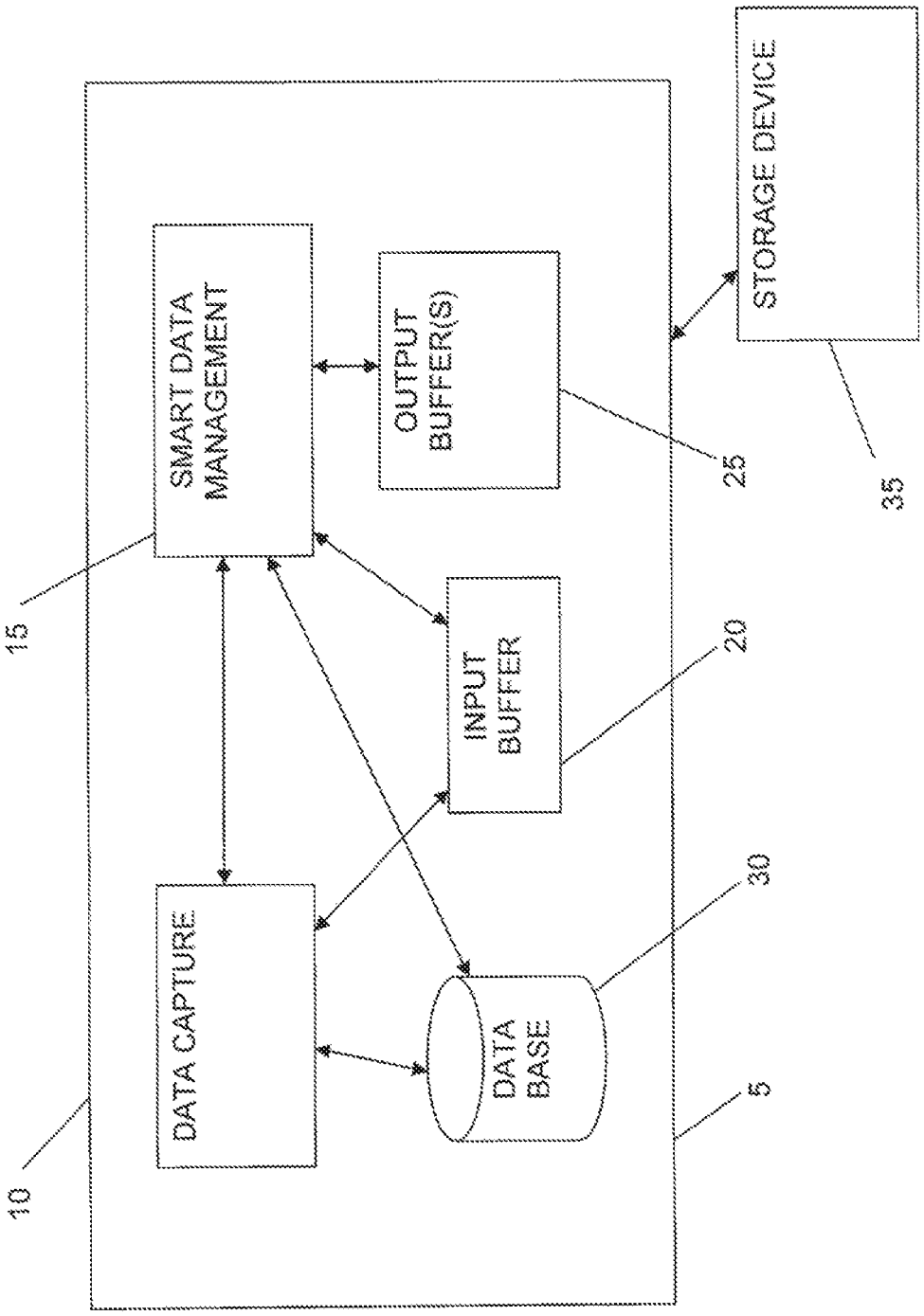

STORAGE DEVICE
SMART DATA MANAGEMENT
OUTPUT BUFFER(S)
INPUT BUFFER
DATA CAPTURE
DATA BASE
10
15
20
25
30
35
5

FIGURE 1

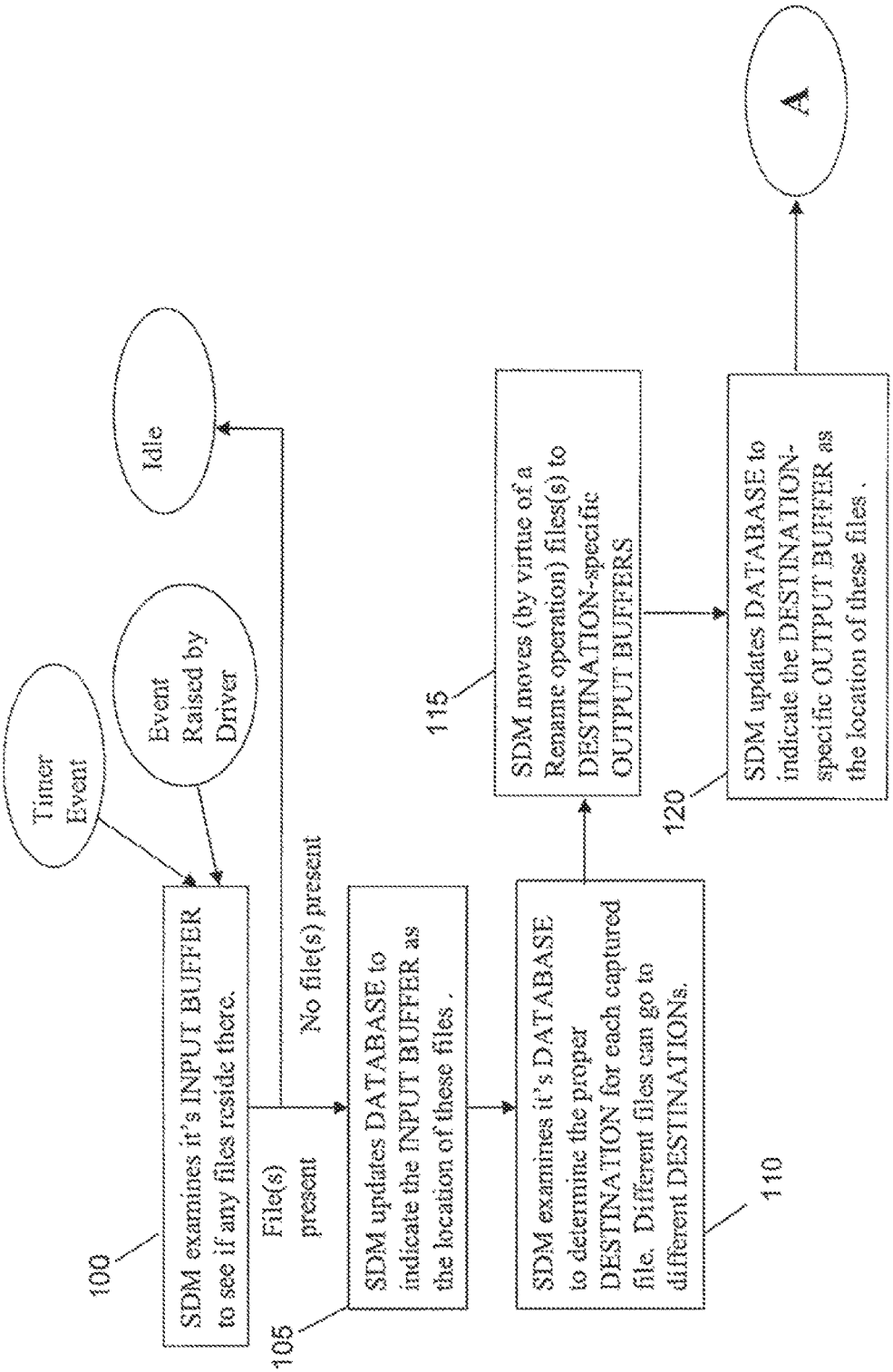

Timer Event
Event Raised by Driver
Idle
100
SDM examines it's INPUT BUFFER to see if any files reside there.
File(s) present
No file(s) present
105
SDM updates DATABASE to indicate the INPUT BUFFER as the location of these files .
SDM examines it's DATABASE to determine the proper DESTINATION for each captured file. Different files can go to different DESTINATIONs.
110
115
SDM moves (by virtue of a Rename operation) files(s) to DESTINATION-specific OUTPUT BUFFERS
120
SDM updates DATABASE to indicate the DESTINATION-specific OUTPUT BUFFER as the location of these files .
A

FIGURE 2A

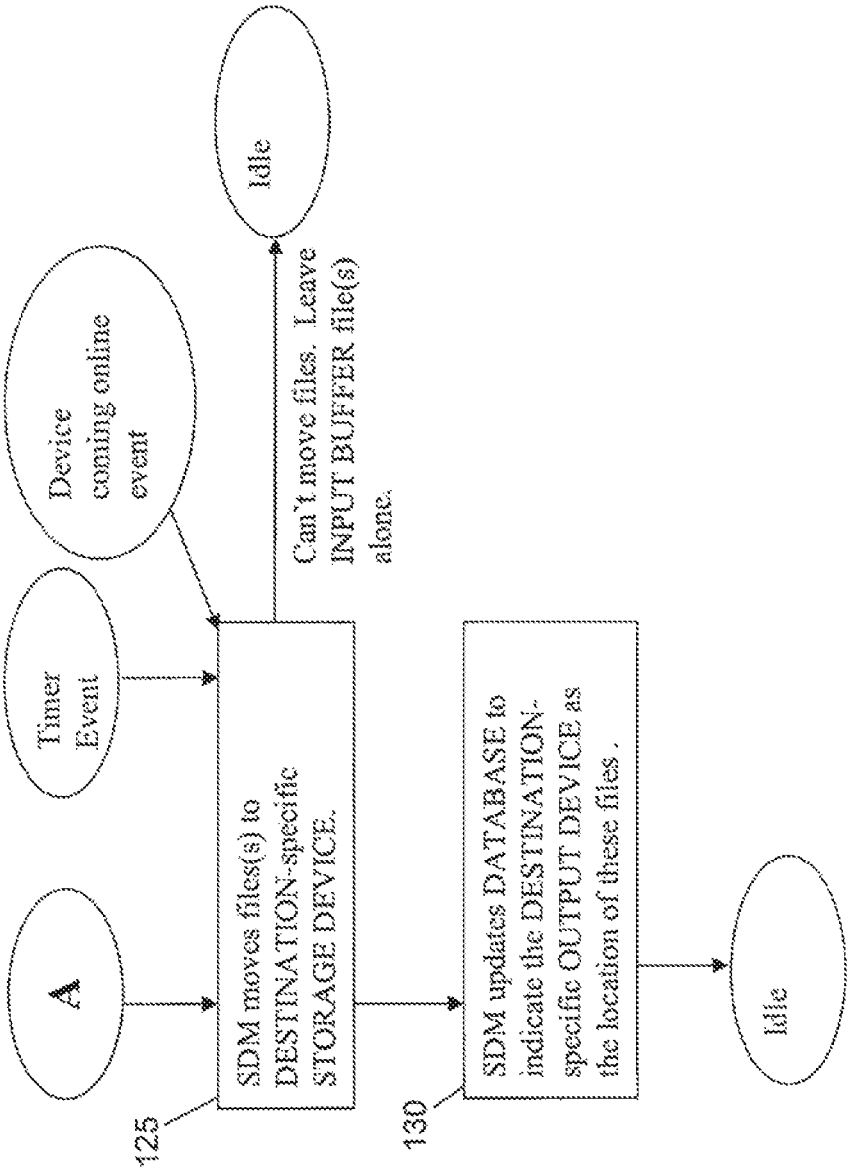

A
Timer Event
Device coming online event
SDM moves files(s) to DESTINATION-specific STORAGE DEVICE.
125
Can't move files. Leave INPUT BUFFER file(s) alone.
Idle
SDM updates DATABASE to indicate the DESTINATION-specific OUTPUT DEVICE as the location of these files .
130
Idle

FIGURE 2B

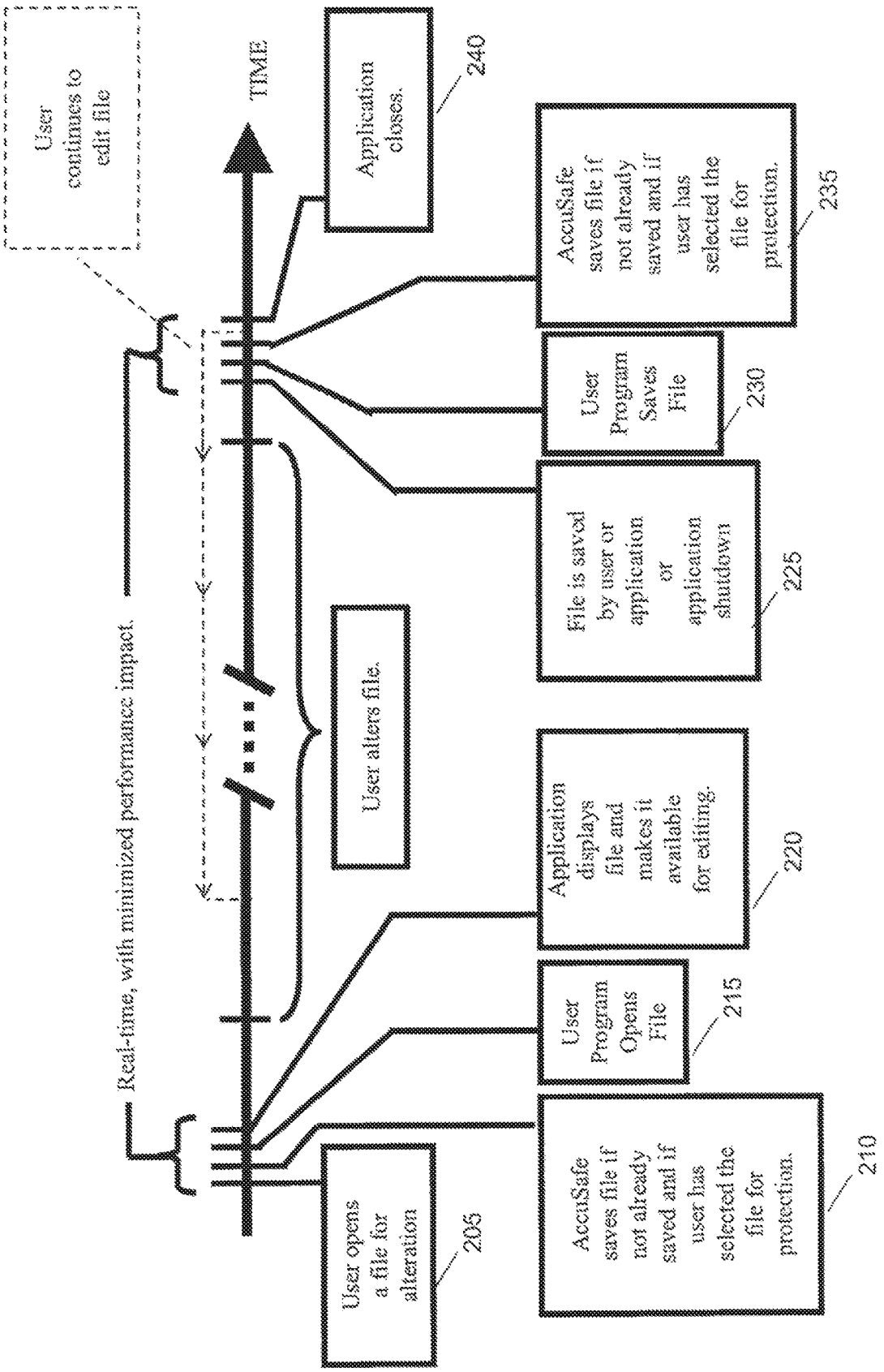

TIME
Real-time, with minimized performance impact.
User continues to edit file
User opens a file for alteration
205
AccuSafe saves file if not already saved and if user has selected the file for protection.
210
User Program Opens File
215
Application displays file and makes it available for editing.
220
User alters file.
File is saved by user or application or application shutdown
225
User Program Saves File
230
AccuSafe saves file if not already saved and if user has selected the file for protection.
235
Application closes.
240

FIGURE 3

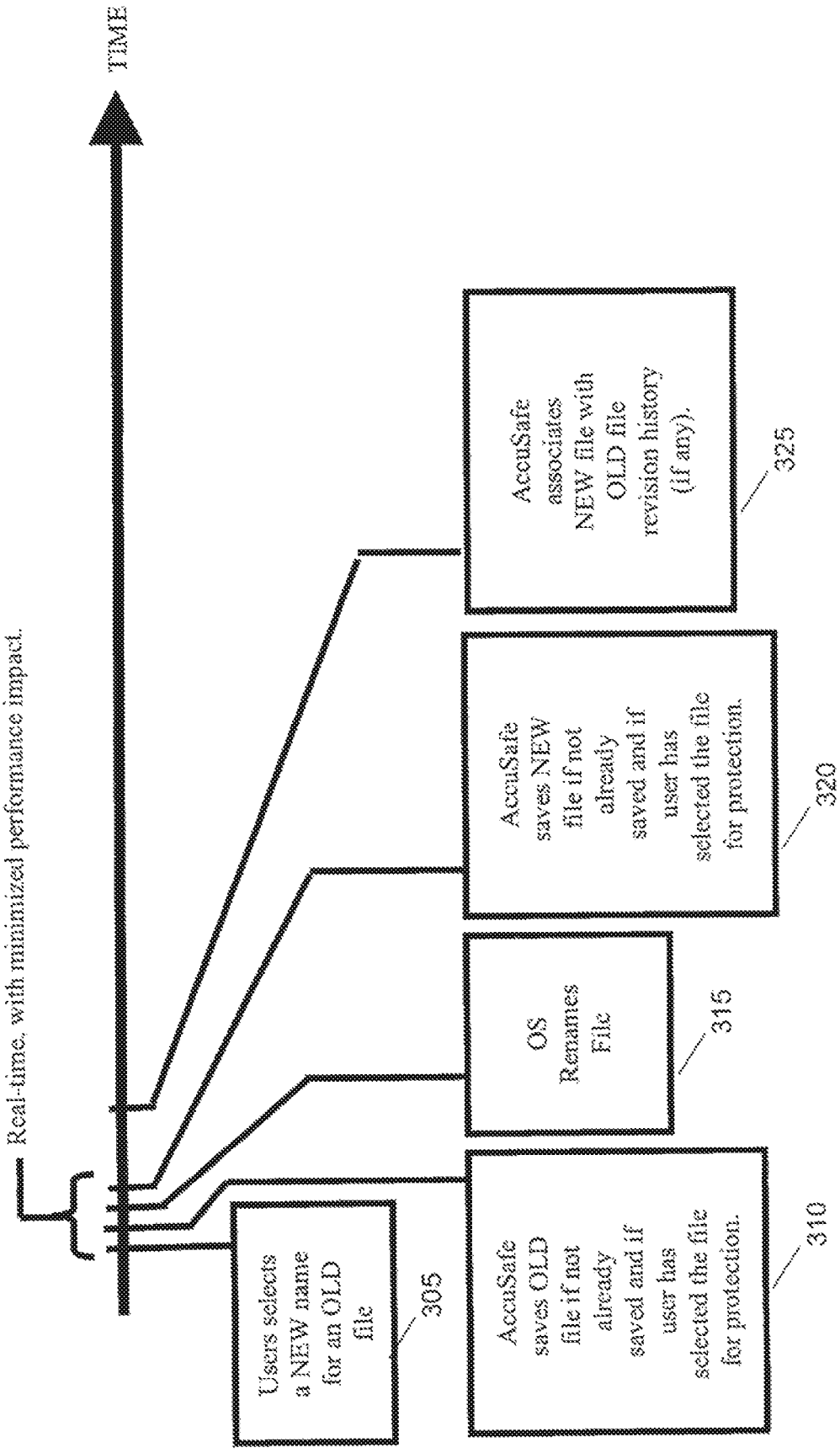

TIME
Real-time, with minimized performance impact.
Users selects a NEW name for an OLD file
305
AccuSafe saves OLD file if not already saved and if user has selected the file for protection.
310
OS Renames File
315
AccuSafe saves NEW file if not already saved and if user has selected the file for protection.
320
AccuSafe associates NEW file with OLD file revision history (if any).
325

FIGURE 4

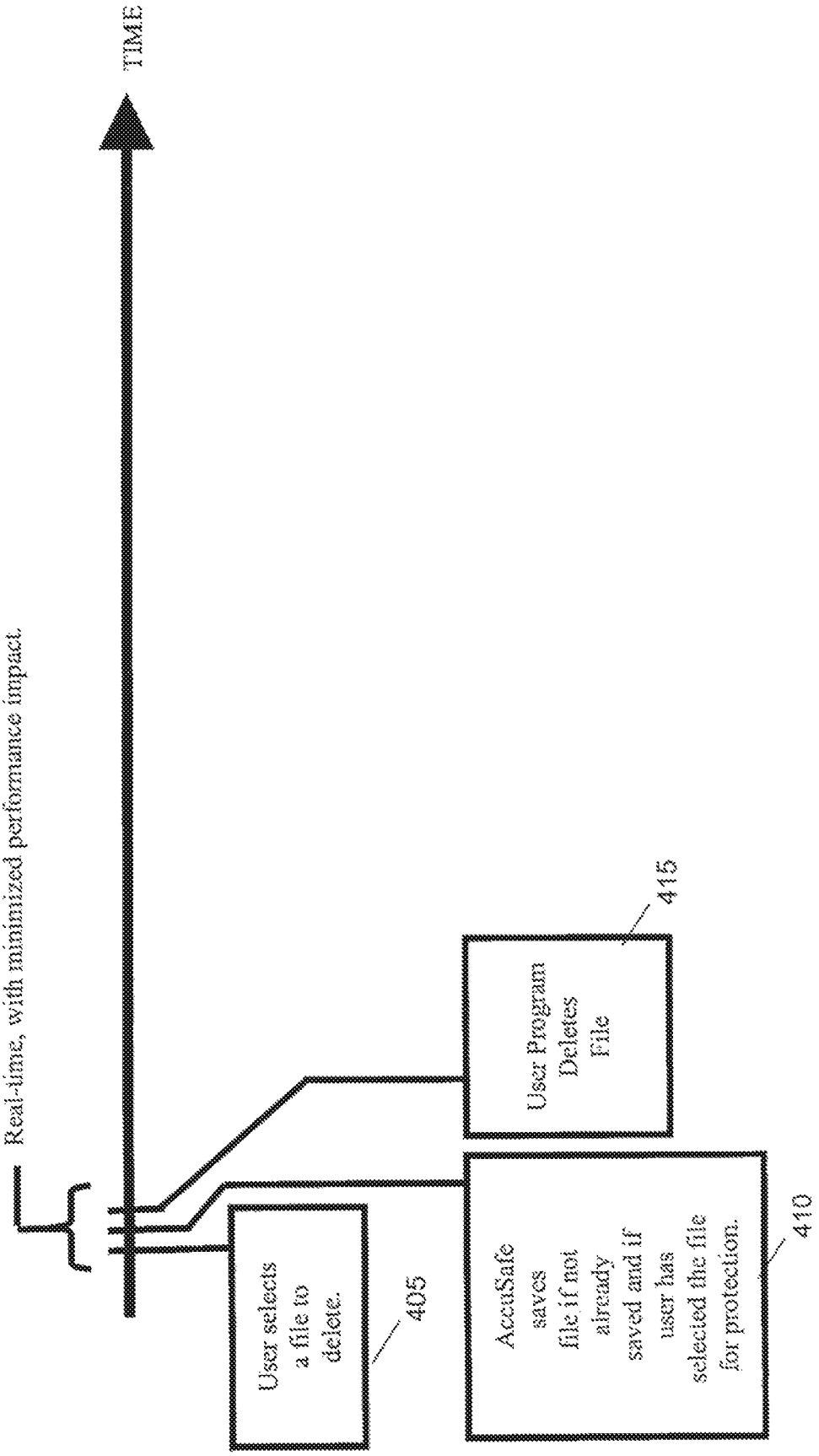

TIME
Real-time, with minimized performance impact.
User selects a file to delete.
405
AccuSafe saves file if not already saved and if user has selected the file for protection.
410
User Program Deletes File
415

FIGURE 5

AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS

This application is a continuation of U.S. patent application Ser. No. 13/925,768 filed Jun. 24, 2013, which is a continuation of U.S. patent application Ser. No. 09/957,459 filed Sep. 21, 2001, now U.S. Pat. No. 8,473,478, which claims the benefit of U.S. Provisional Application No. 60/234,221 filed Sep. 21, 2000, each of which is herein incorporated by reference.

FIELD OF THE INVENTION

This invention relates to file preservation, file capture, file management and file integrity techniques for computing devices.

BACKGROUND OF THE INVENTION

One of the greatest challenges faced by information technology (IT) professionals and computer users today, particularly in the business environment is the protection and management of data. Data may be stored on user workstations, e.g., laptop computers, home or office desktop computers, network servers or other devices external to the workstations. Important data may even be stored on handheld computing devices such as PDAs, PALs and other like devices. Complicating the problem is the fact that the critically of data is increasing and the difficulty of managing it, protecting it from loss and keeping it available is increasing. This is due to a variety of factors, including: 1) the explosion in data volume, particularly that stored on desktop and laptop computers, 2) the increasing complexity of desktop and laptop computer software and hardware and increasing trends toward a paperless environment were absolute reliance (because paper copies are becoming less the norm) on data integrity is increasingly significant.

Many home computer users do not realize the vulnerability of their computer data. Many that do understand the very real potential for data loss, purchase backup systems whose operation and user interface is often confusing and/or time-consuming to use, dramatically decreasing their effectiveness or dependability. As a result, many computer users remain very much at risk of data loss resulting from hardware and/or software failures, fires, stolen equipment, etc. While these risks are significant, the most frequent cause of data loss is user error (accidental file deletes, file overwrites, errant programs, etc.), to which users remain very vulnerable even with most present day backup systems.

The financial impact of information loss is substantial. As reported by the Safeware Insurance Agency, in 1999 alone, insurance claims for damaged, lost and stolen computers (primarily notebook computers) totaled more than $1.9 billion. This figure does not include the untold billions lost in intellectual capital and time. It is costly to recreate lost data and there are significant related costs such as lost productivity and lost opportunity. Consider, for example, the financial and health related impact of a doctor losing all patient contact information and medical histories due to a hard disk crash or some other type of computer failure. In addition, it is costly to keep desktop and laptop computers up and running in the wake of their increasing complexity.

A variety of products have been developed to address data preservation and integrity issues. These products may be loosely grouped into three categories, manual backup systems, schedule based backup systems and mirroring backup systems.

The least efficient and probably one of the most frequently used backup systems is the manual backup. At times determined by the user, the user selects files to be backed up and either utilizes the built in backup procedure for the corresponding application or manually copies the selected files to a desired backup storage media.

The problems with this method of preserving data are self-evident. Backup procedures are often confusing and may differ from application to application. Accordingly, the user must familiarize itself with the various methods for performing backups. In addition, users may forget to backup or elect not to on a given occasion due to time constraints or other reasons. Manual backups often do not allow the user to continue to use the system during the backup procedure. Furthermore, data stored to the backup media is really only a "snapshot" of the data at the time that the backup is performed. Any changes made between manual backups would be lost if there was a failure on the computer's storage device.

Schedule based backup systems typically perform backups according to a schedule either set by the user or preset by the backup software. One of the major disadvantages of each type of schedule-based backup system is that as with manual backups, they miss work done between schedule points. This may cause the user to loose critical information as they work between schedule points. Another disadvantage of schedule-based backups systems is that they are frequently confusing and cumbersome for the user. Still another disadvantage of schedule-based backup systems is that they function poorly if at all when the backup storage device is unavailable, i.e., they cannot be written to due to a communications error or because the device has reached its capacity, is bandwidth limited, or is non-operational for some other reason.

Mirroring is a technique typically applied to disk based backup systems. Mirroring backup systems are the most comprehensive in that everything that happens to the source storage device immediately happens to the backup storage device. That is the backup drive becomes a mirror image of the source drive. Accordingly, if a failure occurs on the source disk, processing can be switched to the backup disk with little or no service interruption.

The strongest advantage of mirroring systems is also their strongest disadvantage. Because there is no operational discrimination, if a file is accidentally deleted from the source disk, it is deleted and cannot be preserved on the backup disk. Likewise, if a virus infects the source disks it is likely to infect the backup disk. Another disadvantage of mirroring systems is that separate backup disks are required for each source disk, doubling the disk requirement for the system. The backup disk must be at least as large as the source disk and the disks must be configured with identical volume mapping. Any extra space that may be present on the backup disk is unavailable.

All of these methods require that the user specify which files/directories to back up, but many users have no concept of flies and directories in their thought process, much less are they able to correlate a particular application (e.g. Microsoft Excel) with the kinds and locations of files they generate. These systems simply require too much user knowledge, and too much user intervention. The backup user's risk increases dramatically the lower his computer knowledge may be.

In view of the foregoing, there is a need for a file capture, preservation and management system that captures files just before and/or just after they have been changed to minimize loss of data between backup events. There is also a need for

file capture and preservation system that captures files even when the destination storage medium for the files is unavailable. There is a further need for a system that allows users to recover easily and quickly from any typo of information loss, including simple user errors, failed software installations or updates, hardware failures (attached storage devices), and lost or stolen laptop computers. Users should be able to recover on their own, without the intervention of the IT staff, and their backup systems should be as "behind the scenes" as possible, requiring little user attention and extremely small amounts of user computer knowledge.

SUMMARY OF THE INVENTION

It is an object of the invention to a file capture, preservation and management method and apparatus that captures files just before and/or just after the files are changed.

It is another object of the invention to provide a file capture, preservation and management method and apparatus that has an imperceptible impact on system performance from the user's point of view.

It is a further object of the invention to provide a file capture, preservation and management method and apparatus that captures and stores files even when there is no connection to the desired storage location.

Still another object of the invention is to provide a file capture, preservation and management method and apparatus that captures and stores files even when the desired storage location is unavailable.

In accordance with an aspect of the invention, a method for archiving files is provided. The method includes, in a computing device, detecting an instruction from a resident program to perform an operation on an operating file. Upon detection of the instruction, capturing the operating file temporally proximate to the operation being performed on the operating file.

In accordance with another aspect of the invention, a method for moving files from a first storage location to a second storage location is provided. The method includes, in a computing device, searching a first storage location for files responsive to the occurrence of a first event and moving the files from the first storage location to the second storage location responsive to a second event.

In accordance with still another aspect of the invention, a method for archiving files is provided. The method includes detecting an instruction from a resident program to perform an operation on an operating file. The method further includes creating an archive file from the operating file and storing the archive file in a first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction. In keeping with the method, the first storage location is searched for an archive file responsive to the occurrence of a first event. The archive file is then moved from the first storage location to the second storage location responsive to a second event.

The accompanying figures show illustrative embodiments of the invention from which these and other of the objectives, novel features and advantages will be readily apparent.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a block diagram of a computing device in accordance with the present invention.

FIG. **2**A is allow chart depicting a process for moving files in accordance with the present invention.

FIG. **2**B is a flow chart showing another process for moving files in accordance with the invention.

FIG. **3** is a time line illustrating a sequence of events in an exemplary operation in accordance with the invention.

FIG. **4** is a time line illustrating a sequence of events in another exemplary operation in accordance with the invention.

FIG. **5** is a time line illustrating a sequence of events in still another exemplary operation in accordance with the invention.

DETAILED DESCRIPTION OF THE EMBODIMENT

Definitions

Operating System (OS)—A computer program that allocates system resources such as memory, disk space, and processor usage and makes it possible for the computer to boot up to a human user interface allowing the user to interact with the computer and control its operation.

Operating File—a system or user file.

Archive File—a file containing all of the data of an operating file in a native or altered format and/or a file containing at least some of the data of an operating file and including references to the location of the remainder of the data of the operating file.

Computing Device—a personal computer, a laptop or notebook computer, a server, a hand-held computing device, a PDA or a PAL. The term computing device is not specific to the kind of operating system being run on such computing device, and includes devices running Microsoft operating systems, Apple Macintosh operating systems, UNIX operating systems, Linux operating systems, and other operating systems.

Storage Location—any storage device, or a buffer, folder, directory or designated area on a storage device.

Personal Attached Storage Device—any internal or external storage device connected to a computing device.

Network Attached Storage Device—any storage device connected directly to a network to which a first computing device is also temporarily or permanently connected, or any storage device connected to a second computing device that is also temporarily or permanently connected to the network to which the first computing device is temporarily or permanently connected.

Internet storage area network—any storage area (device, collection of devices, etc.) that can be accessed by the computing device when the computing device is temporarily or permanently connected to the Internet.

Peer-to-Peer Storage Device—any storage area (device, collection of devices, etc.) that can be accessed by the computing device when it is sharing resources with other network or internet accessible computers.

Resident Program—an operating system (OS) or other program that has control over file operations such as "read", "write", "save", "rename", "delete", "copy", "move", "open", "close", etc.

User Program—an application software program or other computing program installed by the user or by the computer manufacturer for user creation of desired data, documents, or other information that is designed to enhance the functionality and/or enjoyment and/or usability of the computing device. The present invention is directed to an apparatus and/or method for file capture, presentation and management. The invention includes a file capture aspect and smart data management aspect. The invention may be realized as a method and/or an apparatus. More particularly, the invention may be realized as a set of program code instructions

stored on a computer usable medium, a set of program code instructions embodied in a signal for transmitting computer information, and a processor and/or computing device configured as described herein.

FIG. **1** depicts a block diagram in accordance with the present invention comprising a computing device **5** including a file capture block **10** (or file captures), a smart data management block **15** (or smart data manager), an input buffer **20**, output buffer(s) **25**, and a database **30**. A storage device **35** is also provided and may be either internal or external to computing device **5**. The invention functions in conjunction with a resident program on computing device **5**.

In accordance with an embodiment of the invention, file capture block **10** detects an instruction to perform an operation on an operating file initiated by the resident program of computing device **5**. At a moment temporally proximate to when the resident program actually performs the operation, i.e., just before and/or just after the operation is performed on the operating file, or, more preferably, the instant before and/or the instant after the operating file is changed, file capture block **10** captures the operating file or portions thereof. Preferably, the operating file is captured within a few clock cycles of the detection of the instruction.

In keeping with a preferred aspect of the invention, file capture block **10** causes the location of the captured operating file to be recorded in database **30**. The continued process of recording information about captured operating files, or portions thereof, in database **30** creates a record of each version of the operating file, which may be accessed by the user or by other programs.

File capture is preferably executed by creating an archive file from the operating file. The archive file is preferably stored in a temporary storage location, internal or external to the computer, such as input buffer **20**. However, the archive file may be stored directly in storage device **35**. In accordance with a preferred aspect of the invention, storage device **35** may be a personal attached storage device, a network attached storage device, an Internet storage area network, a peer-to-peer storage device, or other storage device.

In keeping with a preferred aspect of the invention, smart data management block **15** manages the migration of the archive file from the input buffer **20** through the output buffers **25** to storage device **35**. This migration may take place either synchronously or asynchronously with the file capture procedures described herein. The time duration from a file arriving in input buffer **20** and when it arrives on archive storage device **35** is managed by the smart data management block **15**. More particularly smart data management block **15** regularly examines input buffer **20** for the presence of archive files. Smart data management block **15** performs this examination upon the occurrence of an event, e.g., messages from the file capture block **10** and/or various messages from the resident program(s), messages from an input buffer timer sent at time intervals controlled by a timer or at time intervals selected by the user. Optionally, smart data management block **15** may then examine database **30** to determine a defined storage location for each of the archive files stored in input buffer **20**. Each archive file stored in the input buffer **20** may be directed to the same storage location or to different storage locations and archive files may be directed to multiple storage locations for redundancy. Preferably, smart data management block **15** moves the archive files to one or more output buffers **25**. More preferably each archive file is moved to output buffers) **25** corresponding to the final storage location(s) for that archive file. Alternatively, all archive files may be moved to a single common output buffer **25** if desired. Upon the occurrence of an event, and/or at defined time intervals, smart data management block **15** moves the archive files from the output buffers **25** to their respective storage device(s) **35**. Exemplary events include but are not limited to messages indicating when storage device **35** is connected and ready for use, messages indicating when storage device **35** is inserted/removed, full, defective, etc., and messages indicating when storage device **35** is disconnected or unavailable, and messages from a storage device timer sent at time intervals controlled by the timer or at time intervals controlled by the user. The input buffer timer and the storage device timer may operate synchronously or non-synchronously.

Under certain conditions, smart data management block **15** may be unable, or may elect not to move the archive files. For example, if storage device **35** is unavailable then smart data management block **15** will not move the archive files to storage device **35**. Among the conditions that may cause storage device **35** to be unavailable are i) storage device **35** is disconnected from computing device **5**, ii) the connection between storage device **35** and computing device **5** is faulty or unacceptably slow, iii) storage device **35** is full, or iv) storage device **35** is malfunctioning. In addition, smart data management block **15** may also regulate movement of archive files according to time schedules set by the user, by monitoring connection bandwidth availability and moving files only during times of high bandwidth availability, or by monitoring other factors Including messages that may received from storage location server requests for archive file transmittal.

A preferred operational mode for smart data management block **15** is illustrated in the flowcharts of FIGS. **2**A and **2**B. In step **100** of FIG. **2**A, smart data manager **15** examines input buffer **20** to determine whether any archive files are stored therein. If no archive files are present, smart data manager **15** rests idle until the next event occurs, if archive files are detected, in step **105**, smart data manager **15** updates database **25** to indicate the location of the archive files; that is, to indicate that the archive files are resident in Input buffer **20**. In step **110**, smart data manager **15** examines database **30** to determine the proper destination for each archive file. In step **115**, smart data manager **15** moves the archive files to output buffers **25**. In step **120**, smart data manager **15** updates database **30** to indicate that the archive files are now stored in the output buffer.

In FIG. **2**B in step **125**, the archive files are moved to one or more storage devices **30**. If smart data manager **15** is unable to move the archive files to any of the storage devices **30**, smart data manager **15** rests idle and does not move the archive files until it is notified that the storage device is available. Accordingly, the archive files remain in either input buffer **20** or output buffer **25** until smart data management block **15** is notified. In step **130** smart data manager **15** updates database **25** to indicate that the archive files are stored in one or more storage devices **30**.

Use Specific to User Program Operations

The following examples are directed to embodiments of the invention specific to operations performed by a user program. The file capture, preservation and management processes of the invention are not limited to execution with the exemplary operation discussed below. The processes of the invention are preferably executed when a resident program causes a change or a change to be imminent in the

operating file. Therefore, the following examples are intended to be exemplary only and non-limiting.

File Capture at File Open

As illustrated in FIG. **3**, in step **205**, the user or a program selects an "open" operation to open an operating file and an instruction to perform that "open" operation on the operating file is sent to the resident program. In step **210**, file capture block **10** detects the instruction and captures the operating file. Optionally, prior to capturing the operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** creates an archive file and stores the archive file in a storage location such as input buffer **20** or storage device **35** just before the resident program opens the operating file. Preferably, file capture block **10** stores the archive file in input buffer **20**. In step **215** the resident program opens the operating file and in step **220** the user program displays the operating file as originally requested, e.g. Microsoft Word, and makes it available for the user to alter, e.g., edit a word processing document, amend or add to a database, etc. Step **210** is performed by momentarily delaying the execution of step **215** in such a manner as to have little or no perceptible impact on system performance from the users point of view.

In step **225**, the user program begins a process to save the altered operating file and an instruction to save the altered operating file is sent to the resident program. In step **230** the resident program saves the altered operating file pursuant to the instruction. In step **235**, immediately after the altered operating file is saved by the resident program, file capture block **10** captures the altered operating file, preferably by creating and storing an archive file of the altered operating file in input buffer **20**. In accordance with a preferred feature of the invention, file capture block **10** may save the archive file in such a way that previous revisions of the operating file are retained. That is, every time the operating file is changed, file capture block **10** saves an archive file and database **30** is updated with information about the archive file. Accordingly, over time, a plurality of archive files may be created from the original operating file. Each archive file represents a revision of the original operating file.

File Capture in the "RENAME" Operation

As illustrated in FIG. **4**, step **305**, in performing an operating file rename operation, the user or a program generates an instruction for the resident program to select a new name for an old operating file. In step **310**, file capture block **10** detects the instruction and captures the old operating file. Optionally, prior to capturing the old operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** creates an archive file of the old operating file and stores the archive file in a storage location such as storage device **35** or, more preferably, input buffer **20** just before the resident program renames the old operating file. In step **315** the resident program renames the old operating file, thus creating a now operating file. Immediately after the old operating file is renamed, file capture block **10** captures the new operating file. Optionally, prior to capturing the new operating file, file capture block **10** may determine whether the new operating file has previously been archived, whether the user has selected the new operating file for protection, or other matching conditions exist. Like the archive file for the old operating file, the archive file for the new operating file is preferably stored in input buffer **20**. In step **325** file capture block **10** and smart data management block **15** associate or link the new operating file with each of the versions of the old operating file to create a continuous operating file revision history.

File Capture in the "Delete" Operation

FIG. **5** illustrates the file capture process in the delete operation. In step **405**, the user or a program identifies an operating file to delete and generates an instruction to the resident program. In step **410**, file capture block **10** detects the instruction and captures the operating file Just before it is deleted in step **415**. Optionally, prior to capturing the operating file, file capture block **10** may check database **30** to a) determine whether the operating file has previously been archived, b) determine whether the user has selected the operating file for protection, or c) determine a match to other defined conditions. If the go-ahead conditions exist, then file capture block **10** preferably captures the operating file. In step **420**, the resident program deletes the operating file.

As shown by the examples given, a clear advantage of the invention is, regardless of the operation being performed, after each file capture step, file capture block **10** preferably updates database **30** to indicate the location of the corresponding archive file. Database **30** may keep track of multiple versions of an operating file, any of which may be accessed at the request of the user or other program.

Another advantage of the invention is that by capturing the operating file just before and/or jest after an operation is performed thereon, the invention achieves near real-time operating file archiving while achieving minimal missed alterations to an operating file.

A further advantage of the invention in its preferred embodiment, is that by intelligently managing the migration of operating files from the input buffer **20** through the output buffer **25** to the storage device **35**, the invention achieves protection of operating files even when the desired storage device is permanently or temporarily unavailable.

INDUSTRIAL APPLICABILITY

The present invention is suited for any application that requires or benefits from near real time file capture, that seeks improved file integrity and/or that seeks efficient management of file storage. For example, the present invention is particularly useful in backup systems, audit trail systems, computer security systems, systems for monitoring computer users and others.

Although the present invention has been described in terms of particular preferred embodiments, it is not limited to those embodiments. Alternative embodiments, examples, and modifications which would still be encompassed by the invention may be made by those skilled in the art, particularly in light of the foregoing teachings.

We claim:

1. A method of restoring a file to a previous version of the file, a current version of the file being available at a local storage location, comprising the steps of:

(A) presenting information for a collection of one or more previous versions of the file, the information for the

collection including information indicative of at least one or more of previous versions of the file, wherein a restorable representation of each version, V, of the previous versions, is retrievable from a remote storage location, the restorable representation having at least information required for recovering the version V by a computational machine with the current version being accessible, the remote storage location being accessible through a network;

(B) responsive to a selection to preview a selected previous version of the file based on the presented information for the collection of (A), presenting a presentable representation of the selected previous version, the selected previous version being one of the previous versions of the file in the presented information for the collection, the presentable representation having at least information required for presenting at least a portion of the selected previous version by the computation machine with the current version being accessible; and

(C) responsive to a selection to restore the selected previous version, retrieving the restorable representation of the selected previous version from the remote storage location and storing the selected previous version by the computational machine as the current version on the local storage location, the selected previous version available from the restorable representation of the selected previous version.

2. The method of claim **1**, wherein, for the presenting of (B), the presentable representation is presented using an application that created the file or the selected previous version.

3. The method of claim **2**, wherein the application is in a read-only mode.

4. The method of claim **1**, wherein the information includes at least two previous versions of the file.

5. The method of claim **1**, wherein the information for the collection includes information indicative of at least one or more of a timestamp, an original file location, and a comment of the previous versions or the file.

6. The method of claim **1**, further comprising the steps of:

(B2) after the presenting of (B) and responsive to a selection to not restoring the previous version and responsive to a second selection of a second previous version of the previous versions, version presenting a second presentable representation of the selected second previous version, the second presentable representation having at least information required for presenting at least a portion of the selected previous version; and

(C2) responsive to a selection to restore the second selected previous version, retrieving the restorable representation of the second selected previous version from the remote storage location and storing the second selected previous version as the current version on the local storage location, the second selected previous version available from the restorable representation of the second selected previous version.

7. The method of claim **1**, wherein the restorable representation of the selected previous version comprises one or more of a compressed and encrypted version of the selected previous version.

8. The method of claim **1**, wherein the restorable representation of the selected previous version comprises an incremental change to the file.

9. The method of claim **1**, wherein the method has a substantially imperceptible impact on system performance from a user's point of view by delaying one or more operations to prevent a perceptible impact on system performance from a user's point of view.

10. A method of restoring a file to a previous version of the file, a current version of the file being available at a local storage location, comprising the steps of:

(A) presenting information for a collection of one or more previous versions of the file, the information for the collection including information indicative of at least one or more of previous versions of the file, wherein a restorable representation of each version, V, of the previous versions, is retrievable from a remote storage location, the restorable representation having at least information required for recovering the version V, the remote storage location being accessible through a network;

(B) responsive to a selection to preview a selected previous version of the file based on the presented information for the collection of (A), presenting a presentable representation of the selected previous version, the selected previous version being one of the previous versions of the file in the presented information for the collection, the presentable representation having at least information required for presenting at least a portion of the selected previous version;

(C) responsive to a selection to restore the selected previous version, retrieving the restorable representation of the selected previous version from the remote storage location and storing the selected previous version as the current version on the local storage location, the selected previous version available from the restorable representation of the selected previous version.

11. The method of claim **10**, wherein, for the presenting of (B), the presentable representation is presented using an application that created the file or the selected previous version.

12. The method of claim **10**, wherein the information includes at least two previous versions of the file.

13. The method of claim **10**, wherein the information for the collection includes information indicative of at least one or more of a timestamp, an original file location, and a comment of the previous versions or the file.

14. The method of claim **10**, further comprising the steps of:

(B2) after the presenting of (B) and responsive to a selection to not restoring the previous version and responsive to a second selection of a second previous version of the previous versions, version presenting a second presentable representation of the selected second previous version, the second presentable representation having at least information required for presenting at least a portion of the selected previous version; and

(C2) responsive to a selection to restore the second selected previous version, retrieving the restorable representation of the second selected previous version from the remote storage location and storing the second selected previous version as the current version on the local storage location, the second selected previous version available from the restorable representation of the second selected previous version.

15. The method of claim **10**, wherein the restorable representation of the selected previous version comprises one or more of a compressed and encrypted version of the selected previous version.

16. The method of claim 10, wherein the restorable representation of the selected previous version comprises an incremental change to the file.

17. The method of claim 10, wherein the method has a substantially imperceptible impact on system performance from a user's point of view by delaying one or more operations to prevent a perceptible impact on system performance from a user's point of view.

18. A method of restoring a file to a previous version of the file, a current version of the file being available at a local storage location, comprising the steps of:

(A) iteratively transmitting a representation of at least a portion of a current version of a file to a remote storage location for archiving the current version responsive to a determination of an availability of the remote storage location for accepting a transmission of the representation, the current version being accessible at a computational machine, the remote storage location being accessible through a network;

(B) presenting information for a collection of one or more previous versions of the file, the information for the collection including information indicative of at least one or more of previous versions of the file, wherein a restorable representation of each version, V, of the previous versions, is retrievable from the remote storage location, the restorable representation having at least information required for recovering the version V by the computational machine, the remote storage location being accessible through a network;

(C) responsive to a selection to preview a selected previous version of the file based on the presented information for the collection of (B), presenting a presentable representation of the selected previous version, the selected previous version being one of the previous versions of the file in the presented information for the collection, the presentable representation having at least information required for presenting at least a portion of the selected previous version by the computation machine with the current version being accessible; and

(D) responsive to a selection to restore the selected previous version, retrieving the restorable representation of the selected previous version from the remote storage location and storing the selected previous version by the computational machine as the current version on the local storage location, the selected previous version available from the restorable representation of the selected previous version.

19. A method of restoring a file to a previous version of the file, a current version of the file being available at a local storage location, comprising the steps of:

(a) receiving a request to display a list of captured revisions for a file;

(b) displaying a list of captured revisions for said file;

(c) receiving a selection from said list indicating a revision of said file to restore;

(d) previewing selected revision of said file, wherein said selected revision of said file is received from an Internet storage area network;

(e) receiving another selection from said list indicating another revision of said file to restore;

(f) previewing selected another revision of said file, wherein said selected another revision of said file is received from a network attached storage location; and

(g) restoring selected another revision of said file following said previewing step in (f), optionally, decrypting said selected revision of said file prior to step (d) and decrypting said selected another revision of said file prior to step (f).

20. The method of claim 19, wherein said method comprises decompressing said selected revision of said file prior to (d) and decompressing said selected another revision of said file prior to (f).

21. The method of claim 19, further comprising the step of decompressing said selected revision of said file prior to step (d) and decompressing said selected another revision of said file prior to step (f).

* * * * *