US 20130212599A1

(19) United States

(12) Patent Application Publication

Giampaolo et al.

(10) Pub. No.: US 2013/0212599 A1

(43) Pub. Date: Aug. 15, 2013

(54) EVENT NOTIFICATION MANAGEMENT

(71) Applicant: **APPLE INC.**, (US)

(72) Inventors: **Dominic Giampaolo**, Mountain View, CA (US); **Eric Weiss**, Paris (FR); **Brent Knight**, Austin, TX (US); **Pavel Cisler**, Los Gatos, CA (US); **Peter McInerney**, Cupertino, CA (US)

(73) Assignee: **APPLE INC.**, Cupertino, CA (US)

(21) Appl. No.: **13/724,148**

(22) Filed: **Dec. 21, 2012**

Related U.S. Application Data

(63) Continuation of application No. 11/499,866, filed on Aug. 4, 2006, now Pat. No. 8,370,853.

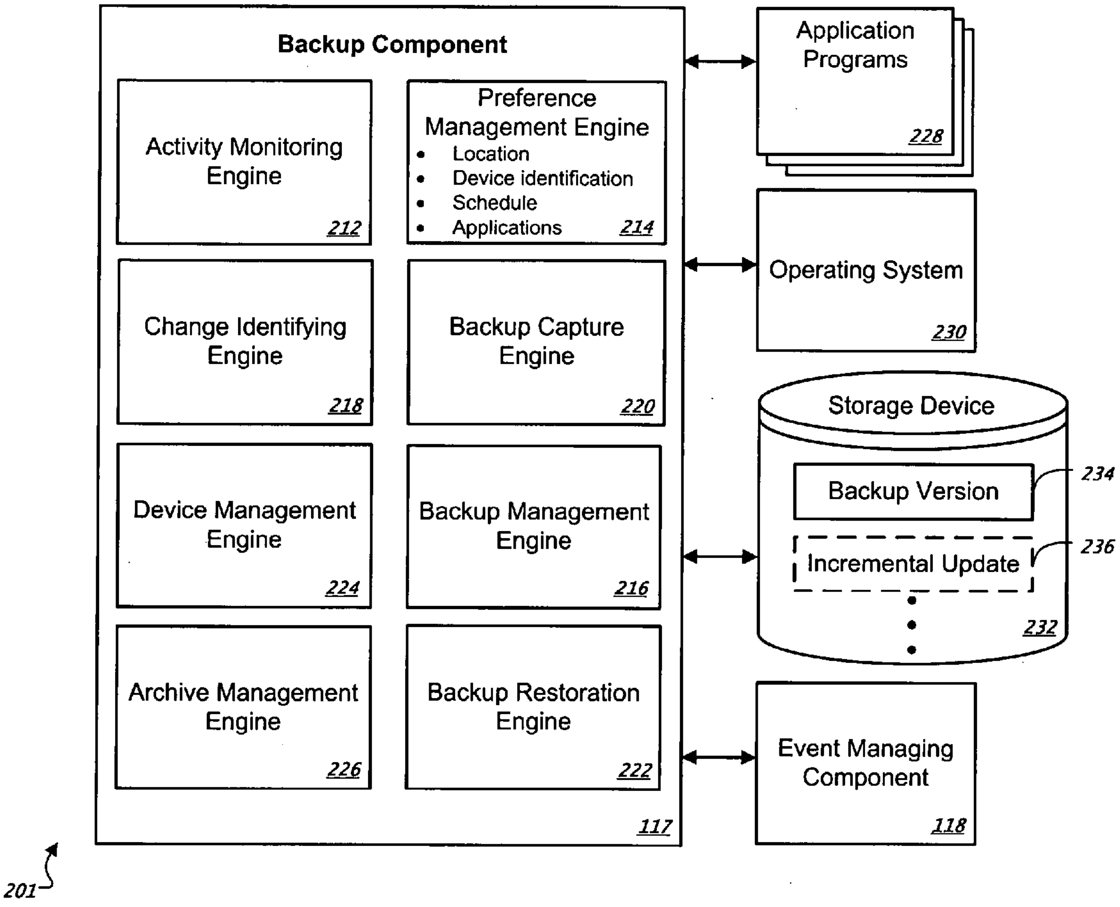
Publication Classification

(51) **Int. Cl.**
G06F 9/54 (2006.01)

(52) **U.S. Cl.**
CPC *G06F 9/542* (2013.01)
USPC **719/318**

(57) **ABSTRACT**

Systems and methods are provided for event notification. In one implementation, a method is provided. A determination is made as to whether a threshold associated with pending event notifications has been exceeded by an incoming event notification. A plurality of pending event notifications that can be combined are identified. Two or more event notifications are combined.



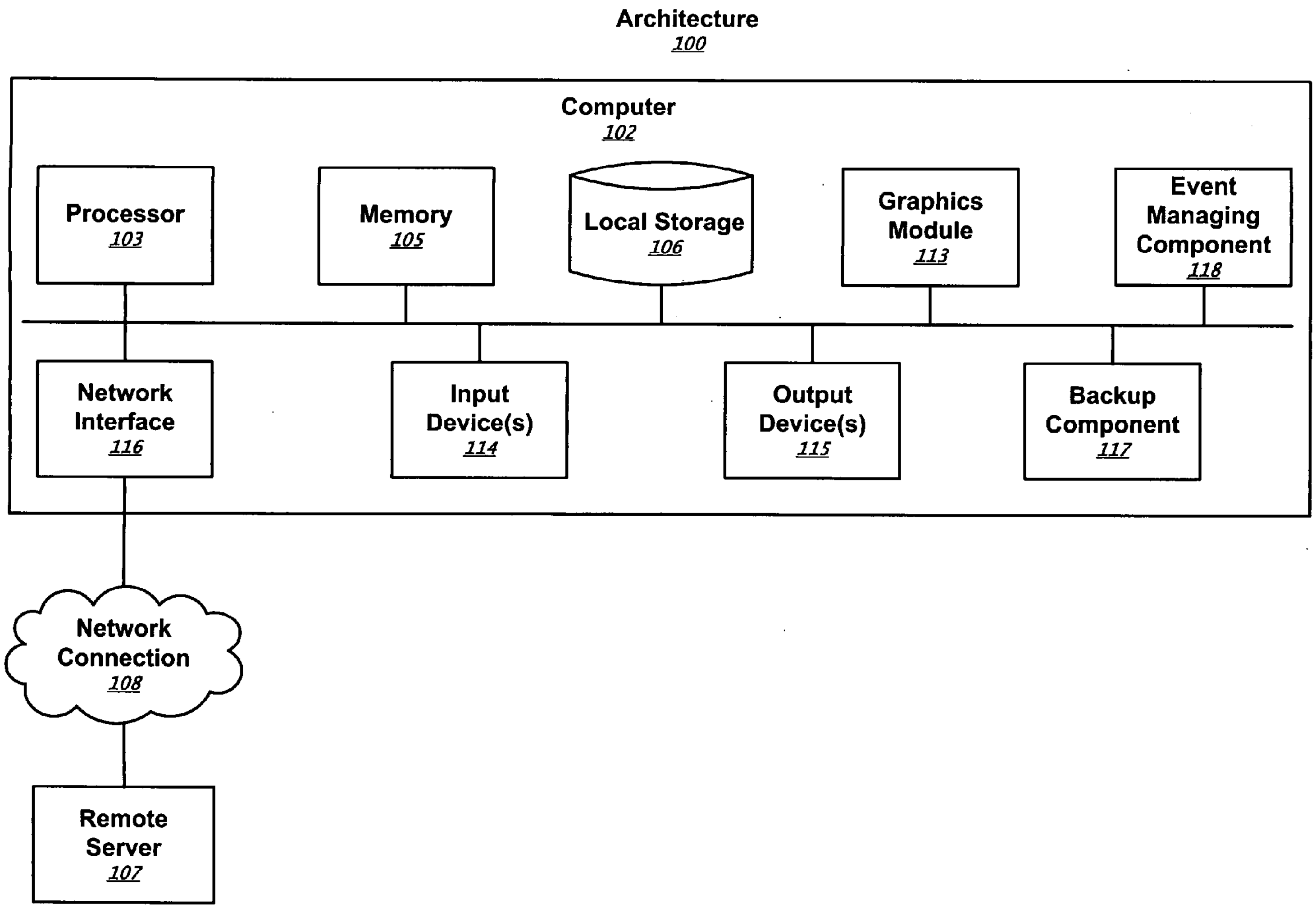

Architecture
100
Computer
102
Processor
103
Memory
105
Local Storage
106
Graphics
Module
113
Event
Managing
Component
118
Network
Interface
116
Input
Device(s)
114
Output
Device(s)
115
Backup
Component
117
Network
Connection
108
Remote
Server
107

FIG. 1

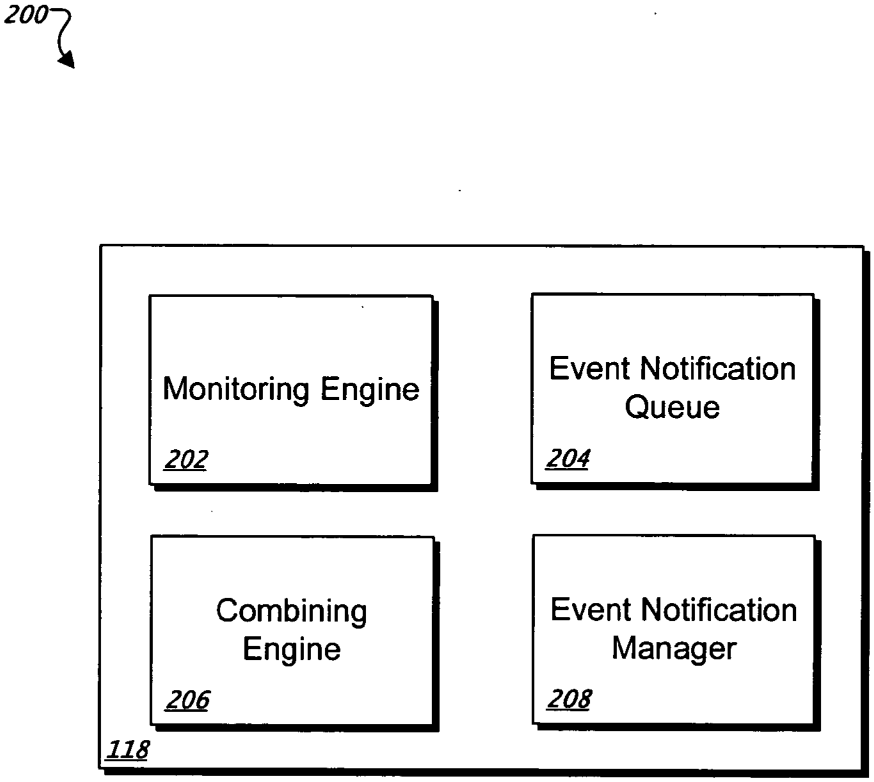

200
Monitoring Engine
202
Event Notification Queue
204
Combining Engine
206
Event Notification Manager
208
118

FIG. 2

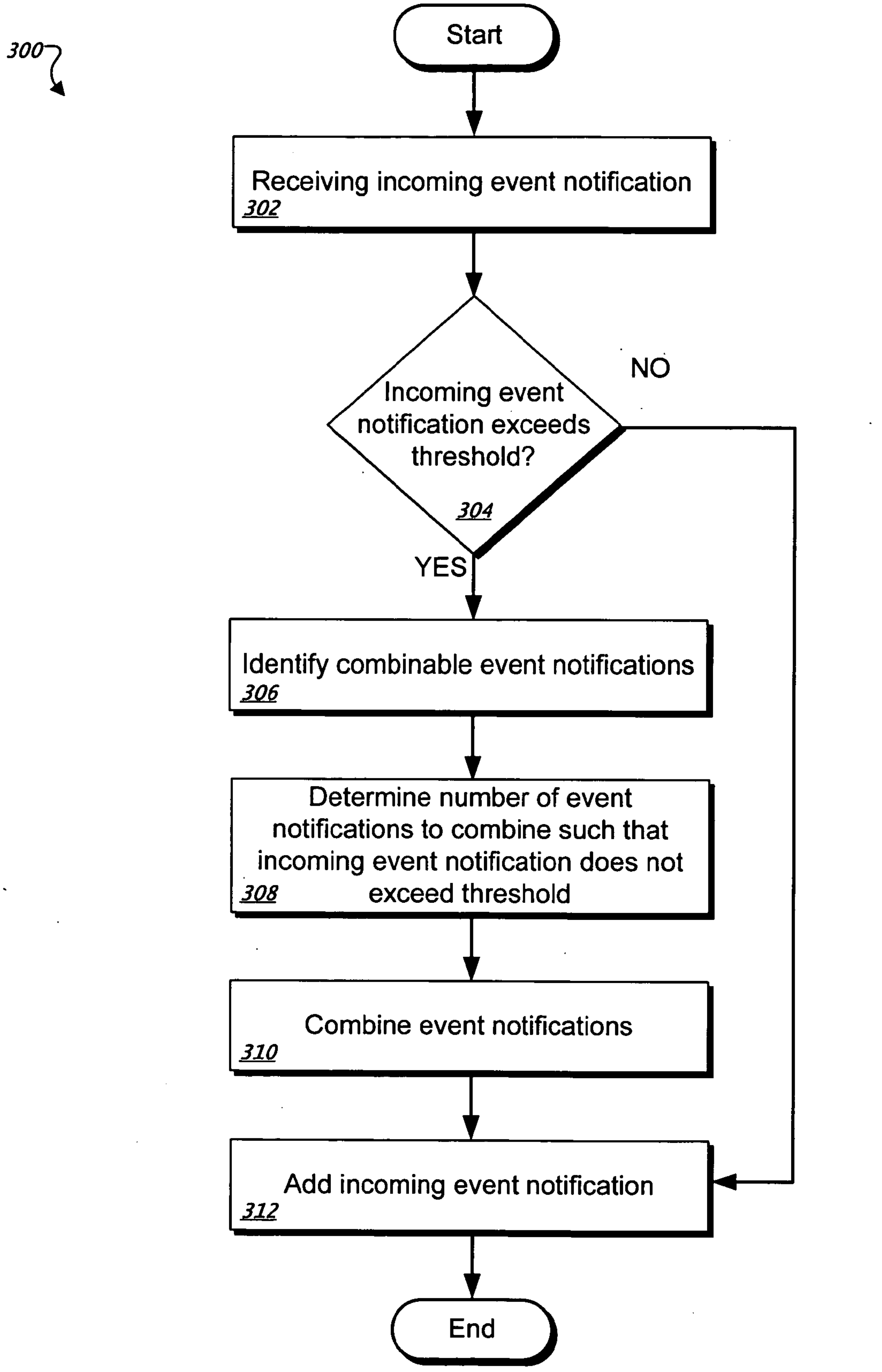

300
Start
Receiving incoming event notification
302
Incoming event notification exceeds threshold?
304
NO
YES
Identify combinable event notifications
306
Determine number of event notifications to combine such that incoming event notification does not exceed threshold
308
Combine event notifications
310
Add incoming event notification
312
End

FIG. 3

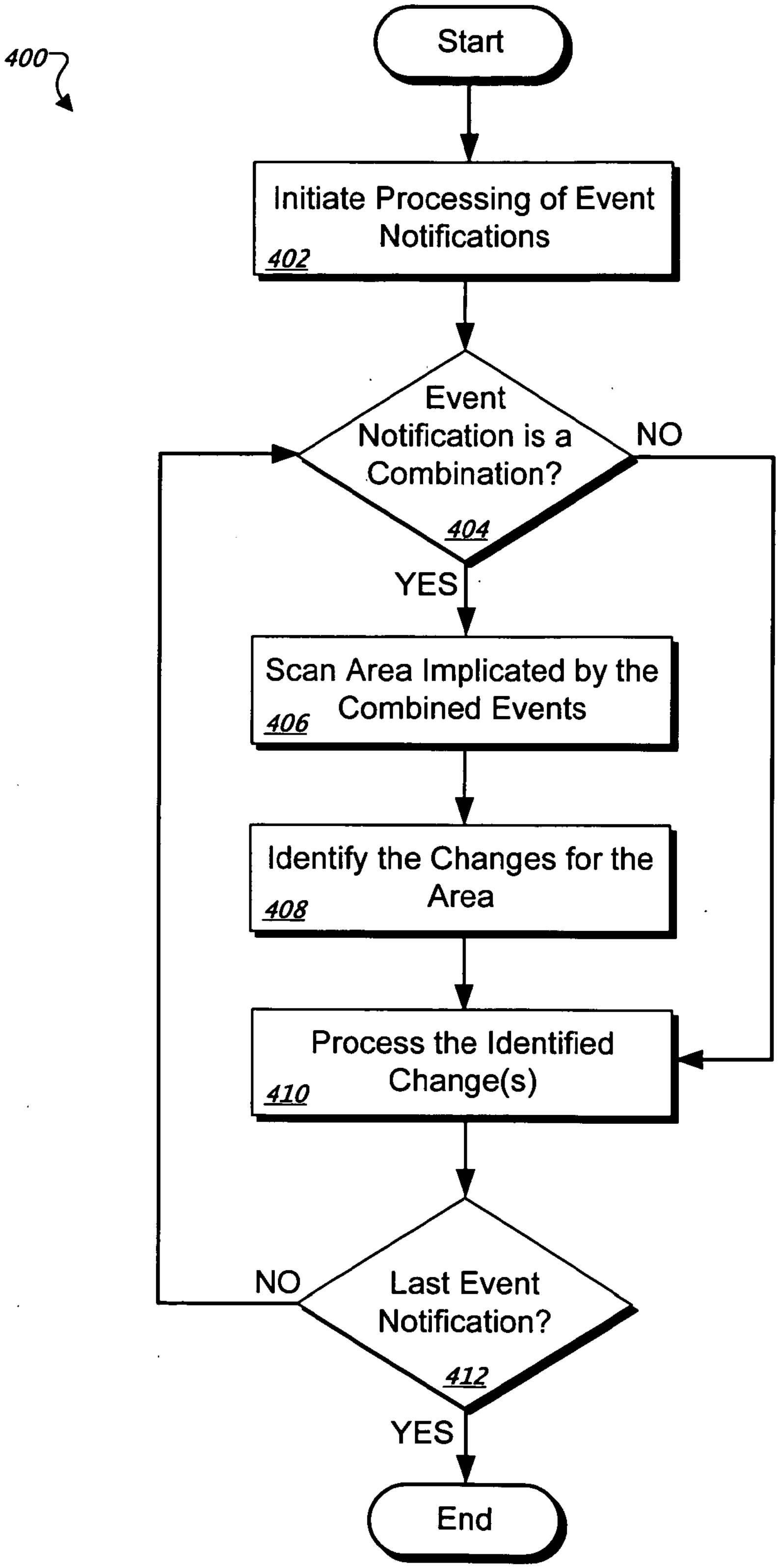

400
Start
Initiate Processing of Event Notifications
402
Event Notification is a Combination?
404
NO
YES
Scan Area Implicated by the Combined Events
406
Identify the Changes for the Area
408
Process the Identified Change(s)
410
NO
Last Event Notification?
412
YES
End

FIG. 4

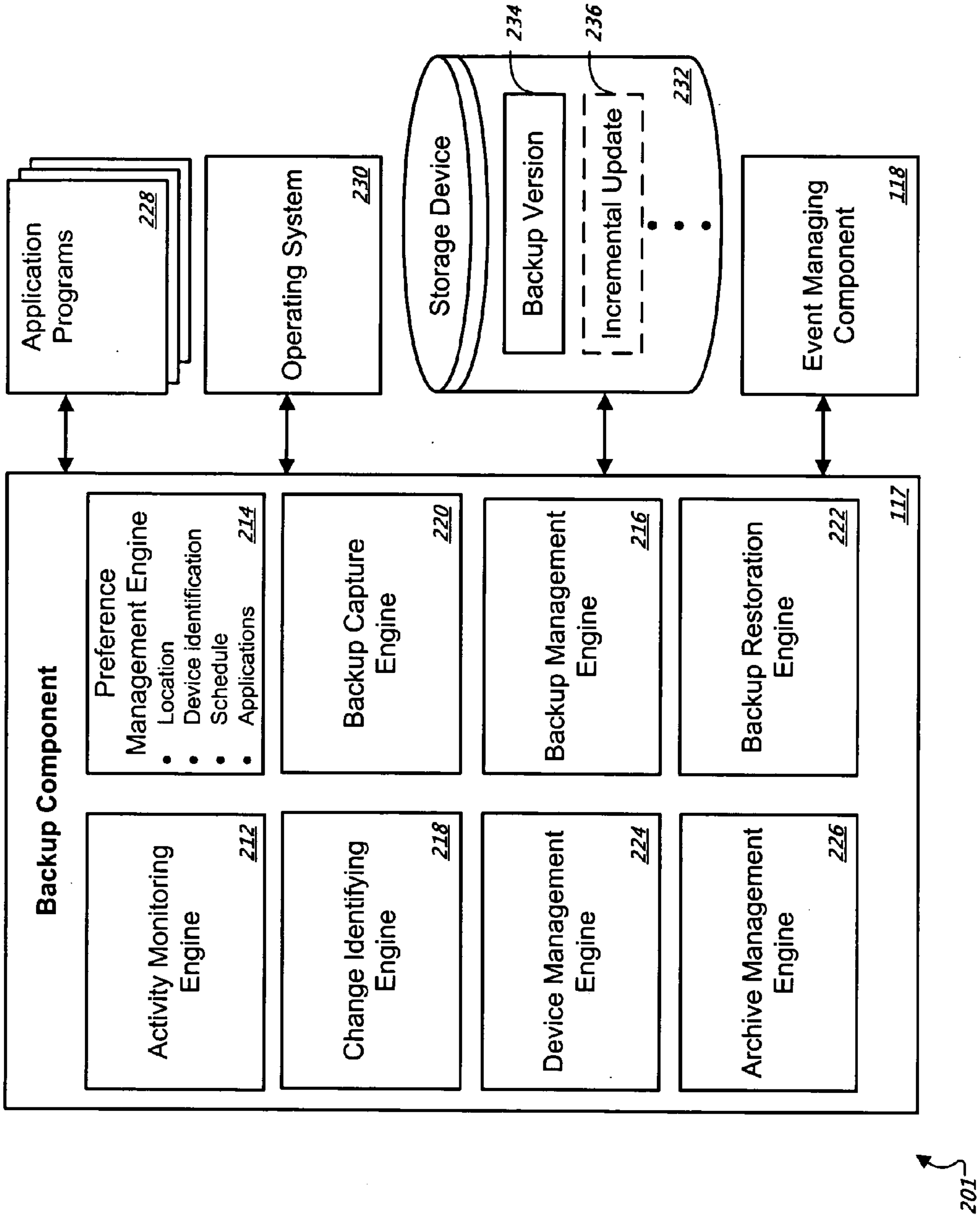
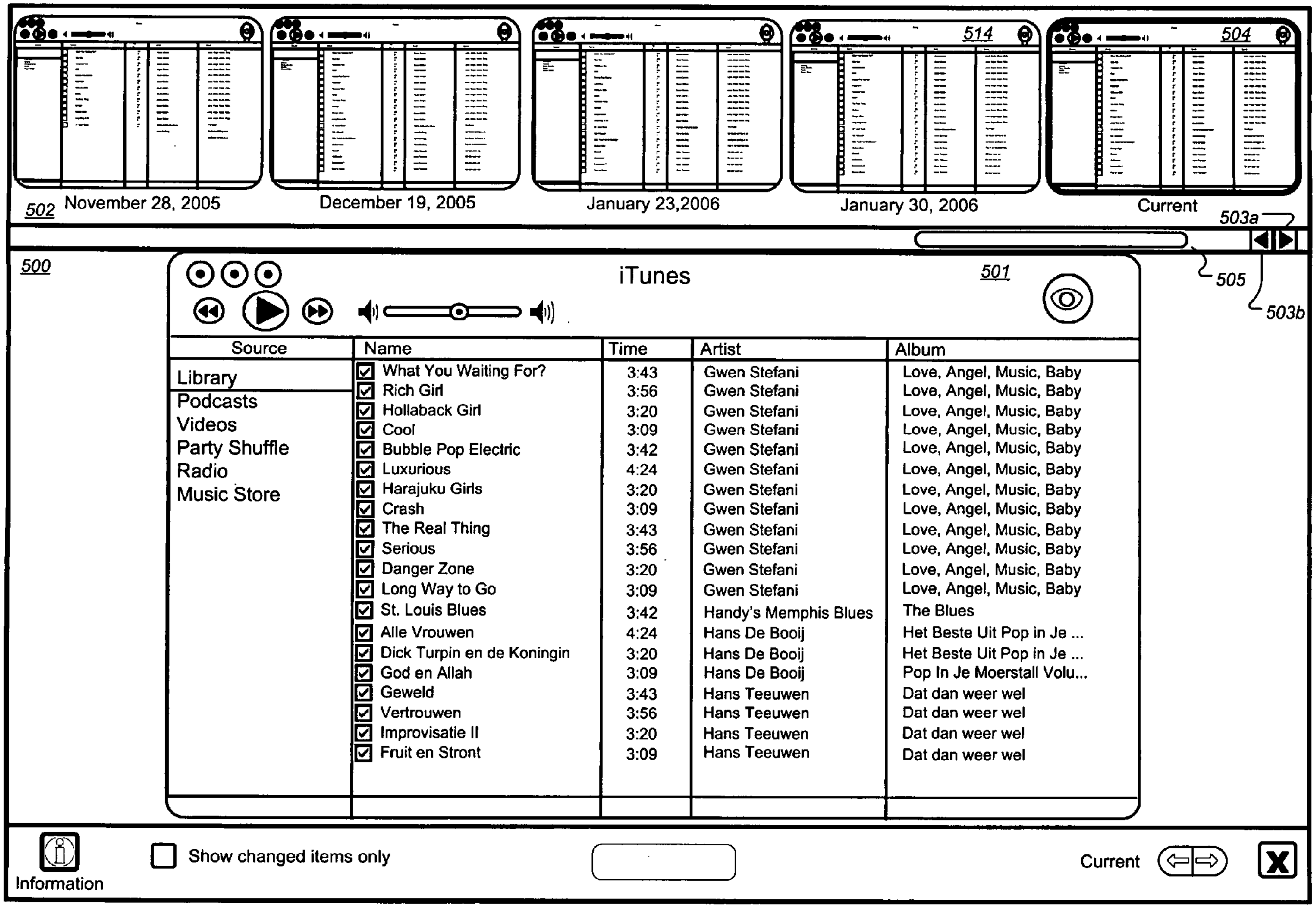

Application Programs
228
Operating System
230
Storage Device
Backup Version
Incremental Update
232
234
236
Event Managing Component
118
Backup Component
117
Preference Management Engine
• Location
• Device identification
• Schedule
• Applications
214
Backup Capture Engine
220
Backup Management Engine
216
Backup Restoration Engine
222
Activity Monitoring Engine
212
Change Identifying Engine
218
Device Management Engine
224
Archive Management Engine
226
201

FIG. 5



☐ Show changed items only

Name	Time	Artist	Album
☑ What You Waiting For?	3:43	Gwen Stefani	Love, Angel, Music, Baby
☑ Rich Girl	3:56	Gwen Stefani	Love, Angel, Music, Baby
☑ Hollaback Girl	3:20	Gwen Stefani	Love, Angel, Music, Baby
☑ Cool	3:09	Gwen Stefani	Love, Angel, Music, Baby
☑ Bubble Pop Electric	3:42	Gwen Stefani	Love, Angel, Music, Baby
☑ Luxurious	4:24	Gwen Stefani	Love, Angel, Music, Baby
☑ Harajuku Girls	3:20	Gwen Stefani	Love, Angel, Music, Baby
☑ Crash	3:09	Gwen Stefani	Love, Angel, Music, Baby
☑ The Real Thing	3:43	Gwen Stefani	Love, Angel, Music, Baby
☑ Serious	3:56	Gwen Stefani	Love, Angel, Music, Baby
☑ Danger Zone	3:20	Gwen Stefani	Love, Angel, Music, Baby
☑ Long Way to Go	3:09	Gwen Stefani	Love, Angel, Music, Baby
☑ St. Louis Blues	3:42	Handy's Memphis Blues	The Blues
☑ Alle Vrouwen	4:24	Hans De Booij	Het Beste Uit Pop in Je ...
☑ Dick Turpin en de Koningin	3:20	Hans De Booij	Het Beste Uit Pop in Je ...
☑ God en Allah	3:09	Hans De Booij	Pop In Je Moerstall Volu...
☑ Geweld	3:43	Hans Teeuwen	Dat dan weer wel
☑ Vertrouwen	3:56	Hans Teeuwen	Dat dan weer wel
☑ Improvisatie II	3:20	Hans Teeuwen	Dat dan weer wel
☑ Fruit en Stront	3:09	Hans Teeuwen	Dat dan weer wel

FIG. 6

EVENT NOTIFICATION MANAGEMENT

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation of U.S. patent application Ser. No. 11/499,866, filed on Aug. 4, 2006, the entire contents of which are hereby incorporated by reference.

TECHNICAL FIELD

[0002] The disclosed implementations relate generally to storing and restoring data.

BACKGROUND

[0003] A hallmark of modern graphical user interfaces is that they allow a large number of graphical objects or items to be displayed on a display screen at the same time. Leading personal computer operating systems, such as Apple Mac OS®, provide user interfaces in which a number of windows can be displayed, overlapped, resized, moved, configured, and reformatted according to the needs of the user or application. Taskbars, menus, virtual buttons and other user interface elements provide mechanisms for accessing and activating windows even when they are hidden behind other windows.

[0004] With the sophisticated tools available, users are encouraged not only to create and save a multitude of items in their computers, but to revise or otherwise improve on them over time. For example, a user can work with a certain file and thereafter save its current version on a storage device. The next day, however, the user could have had second thoughts about the revisions, or could have come up with new ideas, and therefore opens the file again.

[0005] The revision process is usually straightforward if the user wants to add more material to the file or make changes to what is there. But it is typically more difficult for a user who has changed his/her mind about changes that were previously made and wants the file back as it was once before. Application programs for word processing typically let the user "undo" previous edits of a text, at least up to a predefined number of past revisions. The undo feature also usually is configured so that the previously made revisions must be undone in reverse chronological order; that is, the user must first undo the most recently made edit, then the second-most recent one, and so on. If the user saves and closes the document and thereafter opens it again, it may not be possible to automatically undo any previous edits.

[0006] Virtually all computer systems generate event information of some form during operation. For example, "file system events" are used in the Apple Mac OS® X (a corresponding PC version is NT ChangeLog) to provide event notifications for all file system activity (e.g., add file, change file, delete file, etc). Some event information can generate a large data stream, and the capacity to store event notifications can be limited. Additionally, a particular event can require some action by the system and can therefore block the flow of events. To address this situation, event notifications are sometimes dropped to make room for incoming events.

SUMMARY

[0007] Systems and methods are provided for event notification. Events, such as file changes, can be monitored. The event notifications from these events can be used, for example, by a backup system to identify changed data for the next incremental backup operation. The number of event notifications can be limited, leading to dropped events. Dropped events prevent the system from knowing what was changed without searching everything (e.g., searching an entire file system). The loss can be limited by managing the event notifications to combine the event notifications as necessary to maintain space for new incoming events. For example, event notifications for changes to files in a same folder can be combined to a single event notification at a parent level (e.g., a folder that contains the files) to reduce space. Only that parent will need scanned or otherwise examined to determine the changed files instead of the entire file system.

[0008] In general, in one aspect, a method is provided. A determination is made as to whether a threshold associated with pending event notifications has been exceeded by an incoming event notification. A plurality of pending event notifications that can be combined are identified. Two or more event notifications are combined.

[0009] Implementations of the method can include one or more of the following features. Each event notification can identify a change to an entry in a data system. Each event notification can be associated with one or more parent levels of a hierarchical file system and identifying event notifications that can be combined comprises determining two or more event notifications having a common parent level. Combining two or more event notification can include replacing two or more event notifications with a single event notification identifying a change to the common parent level.

[0010] The method can further include scanning the common parent level to identify the combined event notifications and performing a backup operation including backup data associated with the combined event notifications. Scanning the common parent level can include comparing the parent data with data from a previous backup to identify changed data. Receiving an incoming event notification can include receiving an event notification identifying a change to a file system at an event notification queue. The method can further include identifying one or more event notifications as protected, where protected event notifications can not be combined. The method can further include determining a number of event notifications to combine according to one or more criteria. The criteria can include minimizing a number of combined event notifications. Minimizing the number of combined event notifications can include combining a minimum number of event notifications such that the incoming event notification does not exceed the threshold.

[0011] In general, in one aspect, a system is provided. The system includes a monitoring engine for monitoring incoming event notifications. The system includes an event notification queue and an event combiner for combining event notifications in the event notification queue. The system also includes an event notification manager.

[0012] In general, in one aspect, a method is provided. A determination of whether a backup event has occurred is made. Event notifications are evaluated. A backup is generated according to the event notifications. Evaluating event notifications can include identifying an event notification that is a combination of two or more event notifications and scanning data associated with the event notification to identify the combined event notifications.

[0013] Particular embodiments of the subject matter described in this specification can be implemented to realize one or more of the following advantages. The system can combine event notifications in order to reduce dropped event notifications. By combining events, the original event notifications can be more easily identified.

[0014] The details of the various aspects of the subject matter described in this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the invention will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. **1** is a block diagram of an example of an architecture for tracking file system events.

[0016] FIG. **2** is a block diagram of an example of an architecture of an event managing component.

[0017] FIG. **3** is a flow chart of exemplary operations that can be performed to store an event notification.

[0018] FIG. **4** is a flow chart of exemplary operations that can be performed to process event notifications.

[0019] FIG. **5** is a block diagram of an example of an architecture for backing up and restoring application files.

[0020] FIG. **6** is a screen shot showing a time machine interface.

DETAILED DESCRIPTION

[0021] FIG. **1** is a block diagram of an architecture **100** that manages generated event notifications and handles them accordingly. As used herein, an event notification corresponds to activity related to one or more data elements within a computer system. The architecture **100** includes a personal computer **102** communicatively coupled to a remote server **107** via a network interface **116** and a network **108** (e.g., local area network, wireless network, Internet, intranet, etc.). The computer **102** generally includes a processor **103**, memory **105**, one or more input devices **114** (e.g., keyboard, mouse, etc.) and one or more output devices **115** (e.g., a display device). A user interacts with the architecture **100** via the input and output devices **114**, **115**. Architecture **100** as disclosed includes various hardware elements. Architecture **100** can include hardware, software, and combinations of the two.

[0022] The computer **102** also includes a local storage device **106** and a graphics module **113** (e.g., graphics card) for storing information and generating graphical objects, respectively. The local storage device **106** can be a computer-readable medium. The term "computer-readable medium" refers to any medium that includes data and/or participates in providing instructions to a processor for execution, including without limitation, non-volatile media (e.g., optical or magnetic disks), volatile media (e.g., memory) and transmission media. Transmission media includes, without limitation, coaxial cables, copper wire, fiber optics, and computer buses. Transmission media can also take the form of acoustic, light or radio frequency waves.

[0023] While event management is described herein with respect to a personal computer **102**, it should be apparent that the disclosed implementations can be incorporated in, or integrated with, any electronic device that has a user interface, including without limitation, portable and desktop computers, servers, electronics, media players, game devices, mobile phones, email devices, personal digital assistants (PDAs), embedded devices, televisions, other consumer electronic devices, etc.

[0024] Systems and methods are provided for managing events regarding system information, application information or system, application, or user interface state. The systems and methods can be stand alone, or otherwise integrated into a more comprehensive application. In the materials presented below, an integrated system and method for event management is disclosed.

[0025] One of ordinary skill in the art will recognize that the engines, methods, processes and the like that are described can themselves be an individual process or application, part of an operating system, a plug-in, an application, or the like. In one implementation, the system and methods can be implemented as one or more plug-ins that are installed and run on the personal computer **102**. The plug-ins are configured to interact with an operating system (e.g., MAC OS® X, WINDOWS XP, LINUX, etc.) and to perform the various functions, as described with respect to the Figures. A system and method for modifying a user interface view can also be implemented as one or more software applications running on the computer **102**. As used herein, a view refers to an item, element, or other content, capable of being presented in a user interface, that can be subjected to a backup operation by a backup component **117**. For example, a user interface view can contain any number of icons, files, folders, application state information and/or machine state information, etc. Such a system and method can be characterized as a framework or model that can be implemented on various platforms and/or networks (e.g., client/server networks, wireless networks, stand-alone computers, portable electronic devices, mobile phones, etc.), and/or embedded or bundled with one or more software applications (e.g., email, media player, browser, etc.).

[0026] The computer **102** includes a backup component **117** that allows for the storage of versions of the computer's files or other items, for example within the local storage **106** or in an external storage repository. The backup component **117** captures versions of one or more interface views and manages an archive of such backups, for example to facilitate user-initiated restoration based on any of them. As used herein, a view refers to an item, element or other content, capable of being presented in a user interface, that can be subjected to a backup operation by the backup component **117**. For example, a user interface view can contain any number of icons, files, folders, application state information, and/or machine state information, etc. In one implementation, the backup component **117** also allows a user to select any of the stored versions and use it to initiate a restoration of that version in the computer **102**.

[0027] Though discussion is made with reference to a particular user interface view, those of ordinary skill will recognize that such a view can be based on various data structures, files, processes, and other aspects of information management. It follows that modification to file structures, data and the like is also contemplated in order to achieve the modification to the user interface view. In other words, while the restoration of the user interface view from one state to another is the most apparent change from the user's perspective, this is accomplished through the corresponding changes in the underlying system content.

[0028] An event managing component **118** monitors and manages events within the computer **102**. For example, such events can be generated by one or more event logging tools. In one implementation, the event logging tool is a file system event tool such as FS Events (available in products from

Apple Computer in Cupertino, Calif.). The event managing component **118** stores and manages the event notifications as will be described below. For example, the backup component **117** can use a compiled list of events to determine which changed elements should be archived.

[0029] FIG. **2** is a block diagram of an architecture **200** with examples of features that can be included in the event managing component **118**. The event managing component **118** here contains a monitoring engine **202**, an event notification queue **204**, a combining engine **206**, and an event notification manager **208**. An exemplary operation of the event managing component **118** is as follows. The monitoring engine **202** captures a new event that is to be forwarded for storage in the event notification queue **204** until it can be processed. If the event notification manager **208** recognizes that the event notification queue **204** is reaching capacity, the event notification manager **208** can instruct the combining engine **206** to combine two or more of the events into a combined event to save capacity in the event notification queue **204**. For example, if the newly received event relates to a change in a particular directory or folder, that event can be combined with one or more other events relating to the same directory or folder. In some implementations, this causes the information about the specific changes to be lost, but the combined event identifies the localized area (e.g., the directory or folder) that was changed. When the combined event is to be processed, the system can scan the identified directory or folder to identify the changes that were made.

[0030] The monitoring engine **202** detects incoming event notifications. In one implementation, the event notifications originate from a file system activity monitor. An event can occur to a file, folder, data set, system setting, state information, or other individual data element within a computer system. Events include, but are not limited to, element addition, element modification, element deletion, etc. The monitoring engine **202** sends the incoming events for storage processing. If there is currently sufficient capacity in the event notification queue **204**, the detected events remain stored there. In contrast, if the event notification queue **204** currently has insufficient capacity, a check is made to determine whether two or more events can be combined. For example, if the event notification queue **204** is above a set threshold capacity, automatic combining of events can help clear space within the queue.

[0031] When the event notification queue **204** receives an incoming event, it is first determined whether the event notification queue has the capacity to add the new notification. In one implementation, the capacity determination is made by the event notification queue **204** and the results forwarded to the event notification manager **208**. In another implementation, the capacity determination is made by the event notification manager **208** acting on event volume information received from the event notification queue **204**.

[0032] Events can be removed from the event notification queue **204** by processing the events stored within the queue. For example, after the backup component **117** uses the events stored within the event notification queue **204** to create a new archive view, the information within the event notification queue is no longer needed. Events can also be removed from the event notification queue by creating a combined event. In one implementation, all events covered by the scope of the combined event are deleted from the queue upon the addition of the combined event. Thereafter, individual events can be rediscovered by scanning the structure shared by the combined events, such as a parent folder.

[0033] When the event notification queue **204** is at or above capacity or threshold, the combining engine **206** can first attempt to find one or more events within the event notification queue **204** that can be combined with a newly received event. As another example, the combining engine **206** can attempt to find two or more events within the event notification queue **204** that can be combined with each other, to free up resources for handling the newly received event. Thus, in the latter situation, the newly received event is not included in the event combination.

[0034] The combining engine **206** can take into account one or more protection levels in attempting to combine event notifications. That is, one or more event notifications can be identified as protected, meaning that the event notification(s) can not be combined with other event notifications. For example, the protection can be set for those event notifications that can be expected to create worse problems than other event notifications if they are lost before they can be processed.

[0035] If the event notification queue **204** is at or above capacity and there are no combinable events, one or more events can be dropped based on a prioritization scheme. For example, it can be considered critical not to lose a file renaming event, because it can later lead to anomalies and be very difficult to detect. Such critical events can be identified as protected event notifications, which are not combinable. In contrast, a regularly scheduled status update can sometimes be lost without causing significant problems. In one implementation, non-critical events can be dropped in these or other situations. In the later event of queue processing, the backup component **117** will need to scan globally for all element changes in addition to using the critical events contained within the event notification queue **204**.

[0036] The combining engine **206** can be triggered by the event notification manager **208** to combine individual events within the event notification queue **204**. In one implementation, the combining engine is activated when the event notification queue **204** is at maximum capacity. In another implementation, the event notification queue **204** has a threshold level determining when to engage the combining engine **206**.

[0037] In some implementations, the scope of combining activity is based upon system or user settings. For example, the combining engine **206** can begin combining events in no particular order until a queue threshold level has been met. In another implementation, the combining engine **206** can seek out the combinations that are more preferable to make. For example, it could be preferable to combine events that have a high degree of common hierarchical parentage, such as by being related to the same structure in the hierarchy. In contrast, it could be less preferable to combine events that are more unrelated to each other in the hierarchy. This is because the scope and amount of scanning needed in the former case can be less than that necessary in the latter case.

[0038] To create a combination event, the combining engine **206** scans the event notification queue **204** and analyzes the events to determine whether two or more events are combinable. For example, the combining engine **206** can look for elements that share a common feature, such as a parent directory. The combining engine **206** then creates a new combination event regarding the common feature. The combining engine **206** drops the individual events comprising the new combination event from the event notification queue **204** and replaces them with the combination event. In some implementations, the combining engine **206** merges the involved events to form the combined event.

[0039] The event notification manager **208** manages the operation of the monitoring engine **202** and the combining engine **206**. In addition, the event notification manager **208** can monitor the status of the event notification queue **204**. In one implementation, the event notification manager **208** instructs the combining engine **206** to combine events. In one implementation, the event notification manager **208** contains a set of values relating to the trigger events for the combining engine **206**. For example, the set of values can contain a queue threshold value at which combination of events should occur. The set of values can also include a lower threshold value at which the combining engine **206** should cease the combining operation. The event notification manager **208**, in one implementation, triggers the removal of unnecessary event notifications, for example, after the contents of the event notification queue **204** have been processed by the backup component **117**.

[0040] During the processing of events within the event notification queue **204**, the shared feature listed within each combination event can be perused to determine which elements within that shared feature have undergone changes. For example, if the event combination involved two events relating to changes in two images located in the same photo album, the combined event can identify the photo album as a general indication that two or more system events relating to that album have been dropped. In the event processing, then, the album contents can be scanned (or otherwise analyzed) to determine what events relating to the album need to be rediscovered. Upon determining, in this example, that the dropped events related to changes in two of the images, the event managing component **118** can process these rediscovered events similarly to a scenario where they had not been dropped.

[0041] The processing of events can include interaction with the backup component **117**. In the above example, where there had been changes in two images, the event managing component **118** can trigger the backup component **117** to create new backup versions of the altered images.

[0042] In one implementation, the combining engine **206** can also prune events from the event notification queue **204**. For example, the combining engine **206** can scan the event notification queue **204** for multiple events that occurred to the same element, because such events can render each other superfluous or moot, at least in part. For example, adding a new element and then deleting the same element is an example of two events that effectively cancel each other out. In another example, successive additions to a text file can be pruned to a single modification event. Thus, the combining engine **206** can drop one or more events as redundant.

[0043] FIG. **3** is a flow chart of exemplary operations **300** that can be performed in relation to event generation and management. The operations begin in step **302** with receiving an incoming event notification. In one implementation, the event notification can be detected by the monitoring engine **202** (FIG. **2**). The incoming event, in one example, originates from a file system event logging tool.

[0044] In step **304**, the operation determines whether the addition of the incoming event notification exceeds a threshold value associated with the event queue. For example, this determination can be made by the event notification queue **204** (FIG. **2**). In another implementation, the threshold capacity can be detected by the event notification manager **208** (FIG. **2**). If the answer in step **304** is yes, then combinable event notifications are identified in step **306**. For example, event notifications are combinable if they have a shared feature, e.g., a shared parent directory or other hierarchical structure, such that by scanning the shared feature, all of the combinable event elements can be located within that feature. In one implementation, the combining engine **206** (FIG. **2**) locates combinable events. For example, the operation can first attempt to identify any elements that share the same parent structure. If there are no such events, or if the combination thereof does not free up sufficient space in the notification queue, the operation can then move up a hierarchical tree, determining potential combinations among two or more events that share relatively less commonality that those considered earlier.

[0045] Once combinable event notifications have been identified, the number of event notifications to combine can be determined, in step **308**. In one implementation, the operation desires to combine a sufficient number of events such that an incoming event notification will not exceed the queue threshold. The number of event notifications to combine can, in one implementation, be based on default or user settings. For example, in addition to a higher threshold setting that triggers event combination, the event notification manager **208** can contain a lower threshold setting that is the goal for the combining engine **206** when combining events. In one implementation, the order of steps **306** and **308** can be reversed.

[0046] In step **310**, event notifications are combined. For example, the combining engine **206** can combine the most recently received event notification with one or more previously received ones. In one implementation, notification events can be combined with previously combined events, or previously combined events can be combined with each other, to create additional combination events at a higher hierarchical level. In some implementations, a new event notification is generated and used as the combined event, in other implementations one of the events being combined is modified to be the combined event.

[0047] In one implementation, the system can scan (e.g., prior to initiating a backup operation) the particular structure (e.g., folder, directory, or other hierarchical level) that includes combined event notifications in order to determine the underlying events. For example, if two file events were combined into a single folder event notification, the folder is examined to identify which files were the source of the file events that have been lost. In one implementation, the system can compare the folder data with an archive version (e.g., from a previous backup operation) in order to determine which files have changed data, indicating they were the source of the combined file event notifications.

[0048] In step **312**, the incoming event notification is added to the notification queue. When the threshold was not exceeded in step **304**, the newly received notification can be added without modification. In contrast, when the threshold was exceeded in step **304**, the notification that is added in step **312** can be a combined event resulting from the steps **306-310**. In one implementation, the event notification manager **208** adds the incoming event to the event notification queue **204**.

[0049] FIG. **4** is a flow chart of exemplary operations **400** that can be performed in relation to event generation and management. For example, the operations **400** can be performed in processing a queue of event notifications that could include one or more combined event notifications. As has been noted above, in some implementations, when two or more event notifications are combined, the information about the events that triggered them is lost, at least in part. In such situations, for example, the operations **400** can be performed in an attempt to rediscover that missing information so that the events can be processed.

[0050] The operations begin at step **402** with the initiation of the processing of event notifications. For example, this can involve initiating the event managing component **118** (FIG. **1**). As has been mentioned, the event managing component **118** can process event notifications that are stored in a queue.

[0051] In step **404**, it is determined, for each event notification, whether the notification is a combination of two or more event notifications. If so, the operation scans the area implicated by the combined events, in step **406**. For example, when two events relating to the same folder are combined, information about the events can be lost. Although the specifics of the events are no longer known, it is known that they related to a particular folder. The folder can then be considered a hierarchical region that is implicated by these combined events. The system can attempt to rediscover the missing information by scanning the folder. For example, the event managing component **118** can scan each element within the hierarchical region implicated by the combined event, such as by comparing each element's timestamp to the timestamp of the previously generated backup version. In another implementation, the backup component **117** compares the contents of the hierarchical region to the contents of that region within the previously generated backup version to determine the change(s). Other approaches can be used to rediscover the lost event notifications.

[0052] In step **408**, the operation identifies all changed elements within the implicated area. For example, changes can include added data elements, modified data elements, and deleted data elements, etc. The operation then processes the identified change(s) in step **410**. The change(s) processed in step **410** can be from a non-combined event following the determination in step **404**, or from two or more previously combined events that were rediscovered in steps **406-408**. In processing the changes, in one implementation, the backup component **117** can use the list of changed elements to generate a new backup version of each element.

[0053] If, at step **412**, it is found that this is not the last event notification in the list, the operation returns to step **404** to process at least one more event notification. If, instead, this is the last notification to be processed, the operation ends. In one implementation, at the termination of the event processing operation, the event notification manager (**208**) can trigger the cleaning of the event notification queue **204** to clear it of all stored events. The operations **400** can be repeated in part or in whole as required.

[0054] Some operations of the backup component **117** have been described in some of the examples above. There will now be provided an example of features that can be included in the backup component **117**. FIG. **5** is a block diagram of an exemplary architecture **201** for enabling the back up and restoration of data (e.g., application files, application data, settings, parameters or the like), such as those associated with a set of application programs **228**. Backup component **117** provides back up and restoration capability for the system. Many different items or elements can be the subject of a backup operation in the system. For example, folders, files, items, information portions, directories, images, system or application parameters, playlists, e-mail, inbox, application data, address book, preferences, and the like all can be candidates for archiving. Other types are also possible. In this example, the backup component **117** includes at least one storage device **232**. This can be an internal or external resource, or a combination of the two. Any number of local and/or external storage devices can be used by the backup component **117** for storing versions. In this implementation, the backup component **117** views any and all storage device (s) designated for version storage as a single memory bank.

[0055] In one implementation, the backup component **117** runs as a background task on an operating system **230**, such that the task is not visible to the user. The backup component **117** can be capable of running across multiple user accounts.

[0056] The backup component **117** includes an activity monitoring engine **212**. In one implementation, the activity monitoring engine **212** monitors for changes within an application view (e.g. files) that are targeted for a backup operation. A change can also include the addition of new files or data or the deletion of the same. In one implementation, the activity monitoring engine **212** is capable of discerning between a substantive change (e.g. the text within a document has been modified) and a non-substantive change (e.g. the play count within an iTunes playlist has been updated, or several changes cancel each other out) through its interaction with the application programs **228**. The activity monitoring engine **212** can, for example, create a list of modified elements to be used when a backup event is eventually triggered. In one implementation, the activity monitoring engine **212** can monitor the system for periods of inactivity. The activity monitoring engine **212** can then trigger a backup event during a period of time in which the backup operation will not cause a system slowdown for an active user.

[0057] A preference management engine **214** specifies some operating parameters of the backup component **117**. In one implementation, preference management engine **214** contains user-specified and/or system default application parameters for the backup component **117**. These can include settings for the details of capturing and storing the earlier versions. For example, the preference management engine **214** can determine the frequency of the backup capture, the storage location for the backup versions, the types of files, data, or other items that are eligible for backup capture, and the events which trigger a backup capture (periodic or event-driven, etc.).

[0058] In one implementation, the preference management engine **214** can detect that a new storage device is being added to the system and prompt the user whether it should be included as a backup repository. Files and other items can be scheduled for a backup operation due to location (e.g. everything on the C: drive and within D:/photos), a correlation with specific applications (e.g. all pictures, music, e-mail, address book, preferences, and system settings), or a combination of strategies. Different types of items can be scheduled to be stored on different devices or on different segments of a storage device during a backup operation. In one implementation, the backup component **117** stores the versions in a format corresponding to a file system structure.

[0059] A backup management engine **216** coordinates the collection, storage, and retrieval of view versions performed by the backup component **117**. For example, the backup management engine **216** can trigger the activity monitoring engine **212** to watch for activities that satisfy a requirement specified in the preference management engine **214**.

[0060] A change identifying engine **218** locates specific views or other items within to determine if they have changed. The change identifying engine **218** can be capable of discerning a substantive change from a non-substantive change, similar to the example described above for the activity monitoring engine **212**. In one implementation, the change identifying engine **218** traverses a target set of files, data, or other items, comparing a previous version to the current version to determine whether or not a modification has occurred.

[0061] A backup capture engine **220** locates files, data, or other items that are to be backed up. The backup capture engine **220** can invoke the activity monitoring engine **212** and/or the change identifying engine **218**, for example, to generate a capture list. The backup capture engine **220** can then store copies of these elements in one or more targeted storage repositories. The backup capture engine **220** can track multiple version copies of each item included in the backup repository.

[0062] The backup component **117** includes a backup restoration engine **222** to restore previous views (e.g. versions of files, data, or other items). In one implementation, the backup restoration engine **222** provides a user interface (e.g., a graphical user interface) where a user can select the item(s) to be restored.

[0063] A device management engine **224** handles the addition and removal of individual storage devices to be used for archiving views. In one implementation, the preference management engine **214** obtains user settings regarding the identification of individual storage devices for use in archiving. These settings could include, but are not limited to, particular segments of individual devices to use, a threshold capacity which can be filled with archive data, and individual applications to archive to each device. The device management engine **224** records the storage device settings obtained by the preference management engine and uses them to monitor storage device activity. In one implementation, the device management engine **224** can alert the user when a new device has been added to the system. In one implementation, the device management engine **224** can alert the user when an archive-enabled device has been removed from the system. In yet another implementation, the device management engine **224** can alert the user when an archive-enabled device is nearing its threshold storage capacity setting.

[0064] An archive management engine **226** tracks where archived views are being stored. In one implementation, the archive management engine **226** obtains user options from the preference management engine. Such settings can include, but are not limited to, methods to be used to remove older or otherwise unnecessary archived views. These settings can establish criteria for archived view deletion, for instance in the event of storage capacity being reached or on a regular basis. In one implementation, the archive management engine **226** alerts the user when archives are missing because a device has gone offline. In another implementation, the archive management engine **226** bars a user from viewing another user's archived data due to system permissions settings.

[0065] In this example, the first external storage device **232** is being used by the backup component **117** for archiving. The first device **232** contains an initial backup version **234**, which is the first archived view created within this device for a particular item. The first device **232** also contains an incremental update **236**. In one implementation, the incremental update **236** contains links back to data stored within initial backup **234**, such that only one copy of an unchanged piece of data is retained. In this manner, links can also exist between incremental updates. Each incremental update can then contain a copy of each new or changed data item plus a link back to a previously stored copy of each unchanged data item. Any number of incremental updates can exist. If the user changes the scope of data that is being backed up from one incremental update period to another so that the scope of data now includes new data areas, a portion of an incremental update can be considered similar to an initial backup version. Other archive management techniques could be used.

[0066] The backup component **117** can use information stored by the event managing component **118** during incremental backup creation. For example, as described with reference to FIG. **4**, events stored by the event managing component **118** can provide the backup component **117** with a list of changed elements since the last incremental backup event. In some implementations, the backup component **117** includes some or all portions of the event managing component **118**. In another implementation, the event managing component **118** includes some or all portions of the backup component **117**.

[0067] Any number of storage devices can be used by the backup component. In one implementation, different storage devices contain the backup version and incremental updates of data belonging to different applications or to different users on the system. As another example, two or more storage devices can be responsible for backing up contents from separate applications in the system.

[0068] The archived copies can be compressed and/or encrypted. An example of a compression technique is the ZIP file format for data compression and archiving. An example of an encryption technique is the RSA algorithm for public key encryption. Other compression techniques or encryption techniques could be used.

[0069] In one implementation, if multiple users make use of the time machine backup component **117** on a single system, each user can select to keep separate archives. Access to an individual user's archives can be password protected or otherwise held in a secure manner. In one implementation, the archive storage structure mimics a typical file system structure, such that the archived versions can be perused using a standard file system viewing utility.

[0070] It has been mentioned that the backup component **117** can be used for selective restoration of a captured backup version, or a portion thereof. There will now be described with reference to FIG. **6** an example of a user interface that can be generated to facilitate such restoration. The interface is here referred to as a time machine interface **500**, and includes a presentation window **501**, a timeline **502**, and function buttons. The timeline **502** includes a number of snapshots. As used herein, a snapshot refers to a backup element stored in a historical archive that includes a backup of selected items or content as specified by the backup component **117** (FIG. **1**). The time machine engine can be activated to backup or restore media content within the iTunes™ application (available from Apple Computer in Cupertino, Calif.) or another application.

[0071] The time machine user interface **500** can also include function buttons. For example, the interface **500** can include arrow buttons **503***a* and **503***b*, and an associated scroll bar **505** to allow the user to navigate to additional snapshots not shown in the current timeline window. Thus, there can be a large number of snapshots from which to select. As shown, the presentation window **501** is displaying the current state of the iTunes™ application because a "current" snapshot **504** has been selected (highlighted) in the timeline. The current snapshot can be a default selection. The presentation window **501** can show the contents corresponding to the currently selected snapshot, or a portion thereof. Here, the window **501** shows a list of songs that are available in the iTunes™ application, a list sometimes referred to as a playlist.

[0072] In this particular example, the timeline contains a date beneath each snapshot indicating when the snapshot was taken. In some implementations, the user can select items or content within the playlist of any of the snapshots. The selection functionality can be used in earlier snapshots, such as snapshot **514**, to restore missing data to the state associated with the current snapshot **504**.

[0073] The timeline **502** can include a number of snapshots representing earlier versions or states of the iTunes™ library that have been backed up. Each snapshot provides a screenshot representation of the earlier version of the iTunes™ library at a particular point in time. In some implementations, the timeline **502** includes a visual representation of backup elements, such as a miniature version of the earlier state. The timeline can appear across the top portion of the time machine interface **502** (as shown). Alternatively, the timeline does not appear in the top portion of the time machine interface **500** until a user moves their cursor to (or otherwise activates) the timeline (e.g., by activating the top portion of the interface).

[0074] The snapshots within the timeline **502**, in one implementation, were created by the backup component **117** using a list of event notifications generated by the event management component **118** (FIG. **1**). In another implementation, changes occurring to the current playlist are captured by the monitoring engine **202** (FIG. **2**) of the event management component **118**.

[0075] In the case where the event notification queue **204** is at or beyond its threshold value, for example, the event notifications generated through the modification of the contents of the iTunes™ application can be combined by the combining engine **206**. For example, assume that modifications are made to the songs "Rich Girl" and "Crash", both belonging to the album Love, Angel, Music, Baby and that the event notification queue is currently above its capacity. The notifications relating to these modifications can then be combined to create a single event notification regarding the album Love, Angel, Music, Baby. At a later time, when the combined event notification is to be processed, the album Love, Angel, Music, Baby can be scanned to determine what events were dropped, and the corresponding backups can thereafter be performed. Assume, as another example, that modifications are instead made to "Rich Girl" of the album Love, Angel, Music, Baby and to "St. Louis Blues" of the album The Blues, and that the event notifications corresponding to these changes are subsequently combined. Later, upon event processing, the playlist covering both affected albums (and, in some situations, several other albums) can be scanned to determine what events were dropped, for performing the corresponding backup operations. Thus, the latter example can require more extensive scanning than the former.

[0076] In the above description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding. It will be apparent, however, to one skilled in the art that implementations can be practiced without these specific details. In other instances, structures and devices are shown in block diagram form in order to avoid obscuring the disclosure.

[0077] In particular, one skilled in the art will recognize that other architectures and graphics environments can be used, and that the examples can be implemented using graphics tools and products other than those described above. In particular, the client/server approach is merely one example of an architecture for providing the functionality described herein; one skilled in the art will recognize that other, non-client/server approaches can also be used. Some portions of the detailed description are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0078] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0079] An apparatus for performing the operations herein can be specially constructed for the required purposes, or it can comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program can be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0080] The algorithms and modules presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems can be used with programs in accordance with the teachings herein, or it could prove convenient to construct more specialized apparatuses to perform the method steps. The required structure for a variety of these systems will appear from the description. In addition, the present examples are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages can be used to implement the teachings as described herein. Further-more, as will be apparent to one of ordinary skill in the relevant art, the modules, features, attributes, methodologies, and other aspects can be implemented as software, hardware, firmware or any combination of the three. Of course, wherever a component is implemented as software, the component can be implemented as a standalone program, as part of a larger program, as a plurality of separate programs, as a statically or dynamically linked library, as a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future to those of skill in the art of computer programming. Additionally, the present description is in no way limited to implementation in any specific operating system or environment.

[0081] The subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The instructions can be organized into modules (or engines) in different numbers and combinations from the exemplary modules described. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more them. The term "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[0082] While this specification contains many specifics, these should not be construed as limitations on the scope of what may be claimed, but rather as descriptions of features specific to particular implementations of the subject matter. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0083] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0084] The subject matter of this specification has been described in terms of particular embodiments, but other embodiments can be implemented and are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In certain implementations, multitasking and parallel processing may be advantageous. Other variations are within the scope of the following claims.

What is claimed is:

1. (canceled)

2. A method comprising:

determining, by a computer system, that a threshold number of pending event notifications in a queue has been exceeded by an incoming event notification;

determining that two or more event notifications in the queue are associated with a common feature of the computer system; and

combining the two or more event notifications based on the common feature such that the number of pending event notifications, including the combined two or more event notifications, in the queue no longer exceeds the threshold number of event notifications, including replacing two or more event notifications with a single event notification indicating that a change has occurred with regard to the common feature.

3. The method of claim **2**, further comprising:

scanning the computer system to identify the combined event notifications associated with the common feature; and

performing a backup operation including backup data associated with the combined event notifications.

4. The method of claim **3**, where scanning the computer system includes comparing current data related to the common feature with data from a previous backup to identify changed data.

5. The method of claim **2**, where receiving an incoming event notification includes receiving an event notification at an event notification queue that identifies a change to a feature of the computer system.

6. The method of claim **2**, further comprising:

identifying one or more event notifications as protected, where protected event notifications cannot be combined.

7. The method of claim **2**, further comprising:

determining a number of event notifications to combine according to one or more criteria.

8. A non-transitory computer-readable medium including instructions which, when executed by one or more processors causes:

determining that a threshold number of pending event notifications in a queue has been exceeded by an incoming event notification;

determining that two or more event notifications in the queue are associated with a common feature of a computer system; and

combining the two or more event notifications based on the common feature such that the number of pending event notifications, including the combined two or more event notifications, in the queue no longer exceeds the threshold number of event notifications, including replacing two or more event notifications with a single event notification indicating that a change has occurred with regard to the common feature.

9. The non-transitory computer-readable medium of claim **8**, wherein the instructions cause:

scanning the computer system to identify the combined event notifications associated with the common feature; and

performing a backup operation including backup data associated with the combined event notifications.

10. The non-transitory computer-readable medium of claim **9**, where the instructions that cause scanning the computer system include instructions that cause comparing current data related to the common feature with data from a previous backup to identify changed data.

11. The non-transitory computer-readable medium of claim **8**, where the instructions that cause receiving an incoming event notification include instructions that cause receiving an event notification at an event notification queue that identifies a change to a feature of the computer system.

12. The non-transitory computer-readable medium of claim **8**, where the instructions cause:

identifying one or more event notifications as protected, where protected event notifications cannot be combined.

13. The non-transitory computer-readable medium of claim **8**, where the instructions cause:

determining a number of event notifications to combine according to one or more criteria.

14. A system comprising:

one or more processors; and

a non-transitory computer-readable medium including instructions which, when executed by the one or more processors causes:

determining that a threshold number of pending event notifications in a queue has been exceeded by an incoming event notification;

determining that two or more event notifications in the queue are associated with a common feature of a computer system; and

combining the two or more event notifications based on the common feature such that the number of pending event notifications, including the combined two or more event notifications, in the queue no longer exceeds the threshold number of event notifications, including replacing two or more event notifications with a single event notification indicating that a change has occurred with regard to the common feature.

15. The system of claim **14**, wherein the instructions cause:

scanning the computer system to identify the combined event notifications associated with the common feature; and

performing a backup operation including backup data associated with the combined event notifications.

16. The system of claim **15**, where the instructions that cause scanning the computer system include instructions that cause comparing current data related to the common feature with data from a previous backup to identify changed data.

17. The system of claim **14**, where the instructions that cause receiving an incoming event notification include instructions that cause receiving an event notification at an event notification queue that identifies a change to a feature of the computer system.

18. The system of claim **14**, where the instructions cause:

identifying one or more event notifications as protected, where protected event notifications cannot be combined.

19. The system of claim **14**, where the instructions cause:

determining a number of event notifications to combine according to one or more criteria.

* * * * *