US 20140337608A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2014/0337608 A1

(43) Pub. Date: Nov. 13, 2014

LEE

(54) METHOD AND SYSTEM FOR BOOTING AND AUTOMATICALLY UPDATING SOFTWARE, AND RECOVERING FROM UPDATE ERROR, AND COMPUTER READABLE RECORDING MEDIUM STORING METHOD

(71) Applicant: **SAMSUNG ELECTRONICS CO., LTD.**, Suwon-si (KR)

(72) Inventor: **Jong-suk LEE**, Suwon-si (KR)

(21) Appl. No.: **14/341,428**

(22) Filed: **Jul. 25, 2014**

Related U.S. Application Data

(60) Continuation of application No. 13/291,901, filed on Nov. 8, 2011, now Pat. No. 8,825,995, which is a division of application No. 11/513,057, filed on Aug. 31, 2006, now Pat. No. 8,166,285.

(30) **Foreign Application Priority Data**

Sep. 27, 2005 (KR) 10-2005-0089692

Publication Classification

(51) **Int. Cl.**

G06F 9/445 (2006.01)
G06F 11/14 (2006.01)
G06F 9/44 (2006.01)

(52) **U.S. Cl.**

CPC ***G06F 8/65*** (2013.01); ***G06F 9/4401*** (2013.01); ***G06F 11/1417*** (2013.01); ***G06F 11/1433*** (2013.01); *G06F 2201/865* (2013.01)

USPC **713/1**; 717/168

(57) **ABSTRACT**

Provided are a booting method of updating software components installed in a system and recovering from an error that occurs in an update, a method and system for automatically updating the software and recovering from the error, and a computer readable recording medium storing the method. The master boot record and the backup boot record are used to stably update a kernel and effectively recover from an update error. The component configuration database is used to update a plurality of software components including a kernel in a transaction, and perfectly recover from an update error.

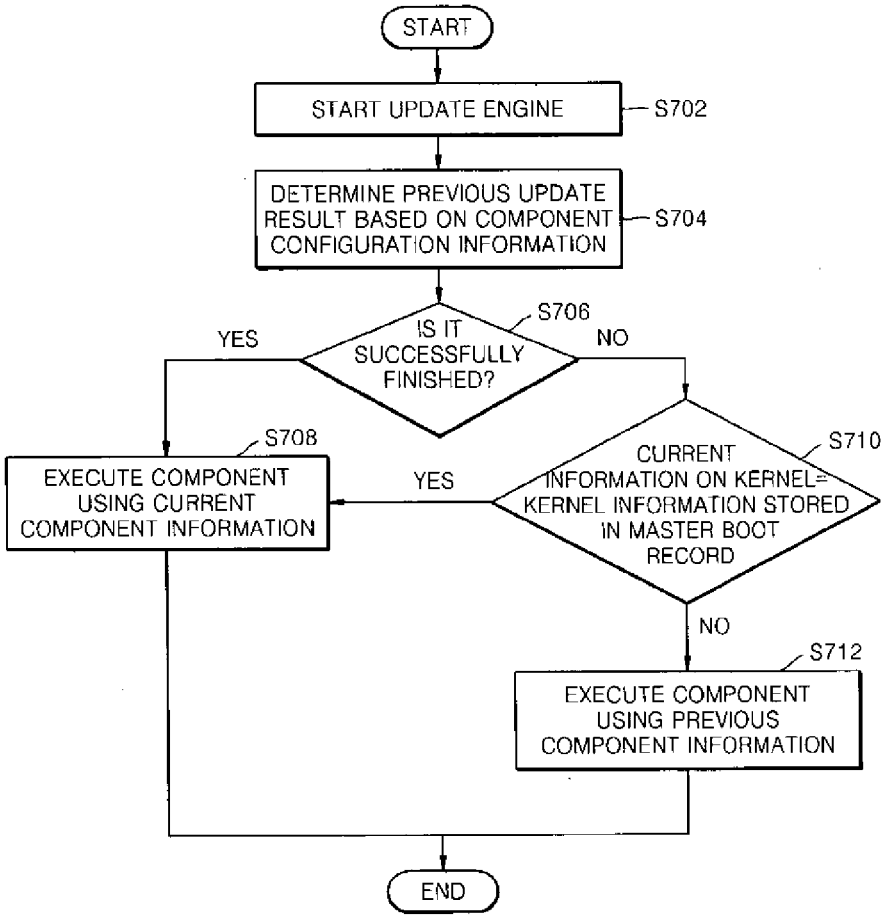


FIG. 1 (RELATED ART)

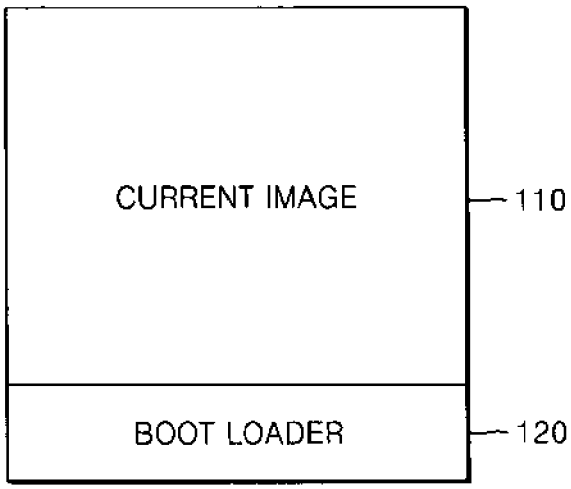


FIG. 2 (RELATED ART)

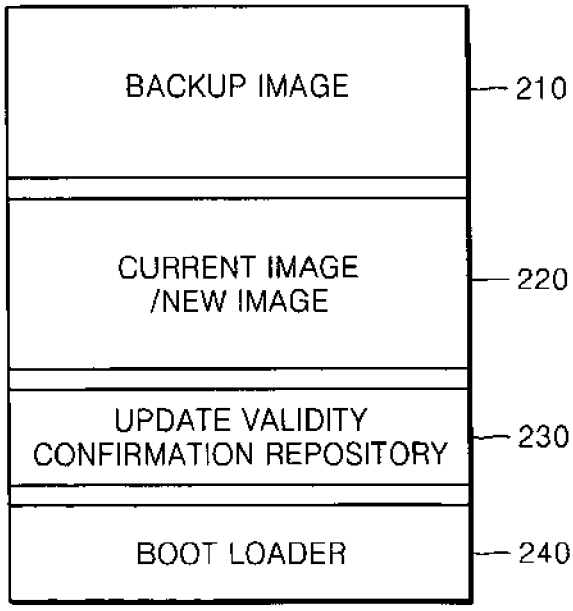


FIG. 3 (RELATED ART)

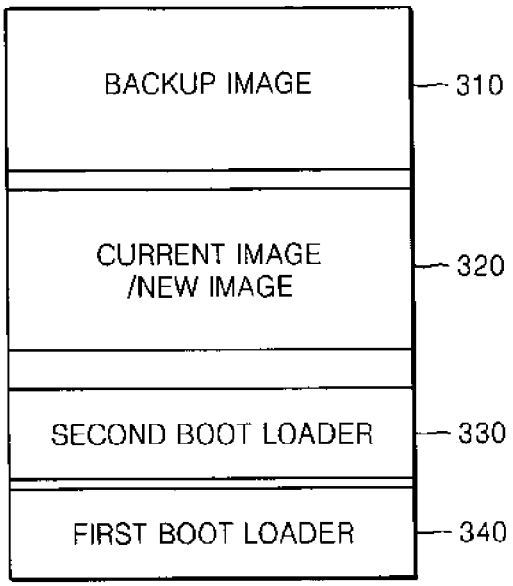


FIG. 4A

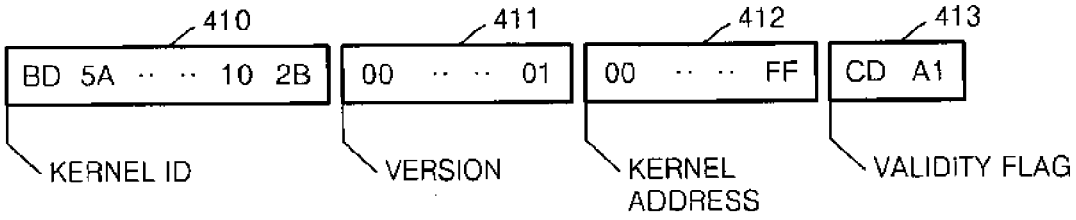


FIG. 4B

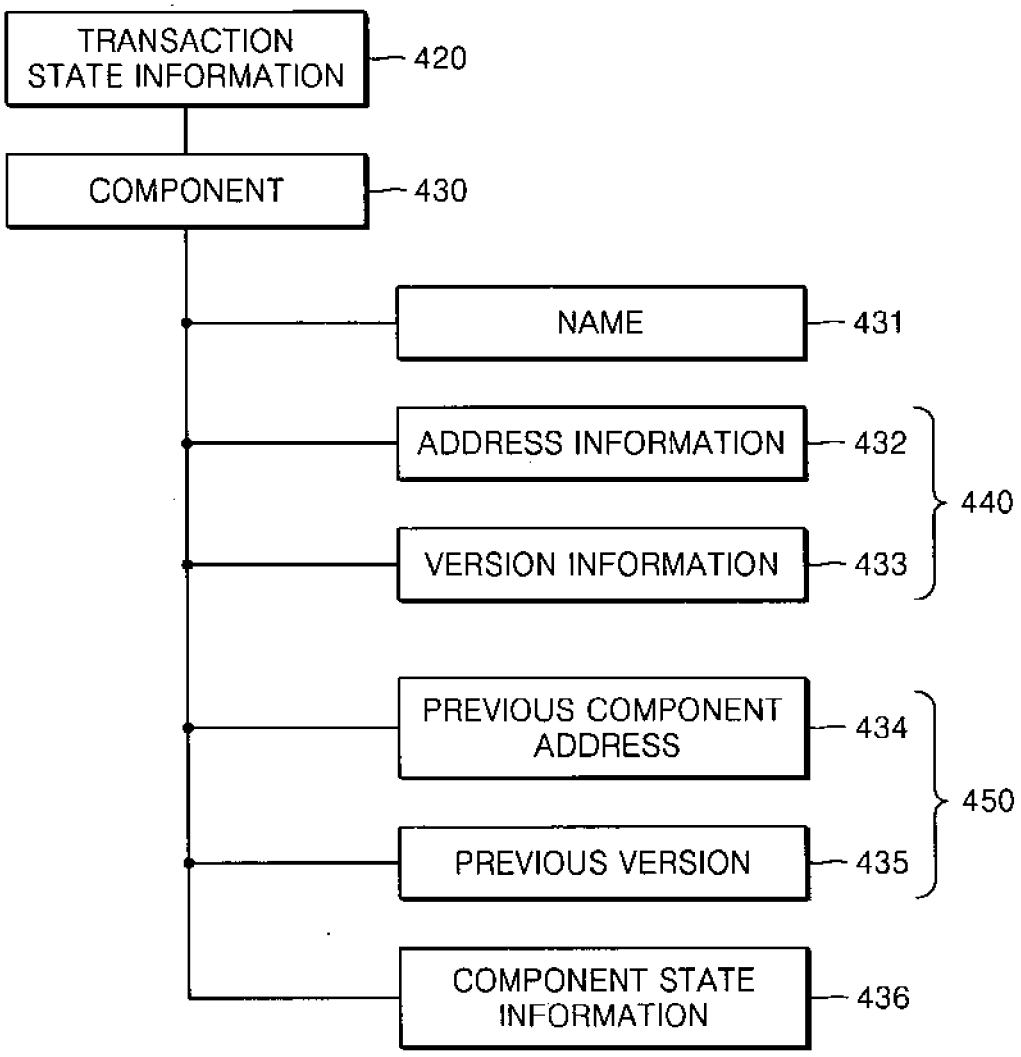

TRANSACTION STATE INFORMATION
420
COMPONENT
430
NAME
431
ADDRESS INFORMATION
432
VERSION INFORMATION
433
440
PREVIOUS COMPONENT ADDRESS
434
PREVIOUS VERSION
435
450
COMPONENT STATE INFORMATION
436

FIG. 5

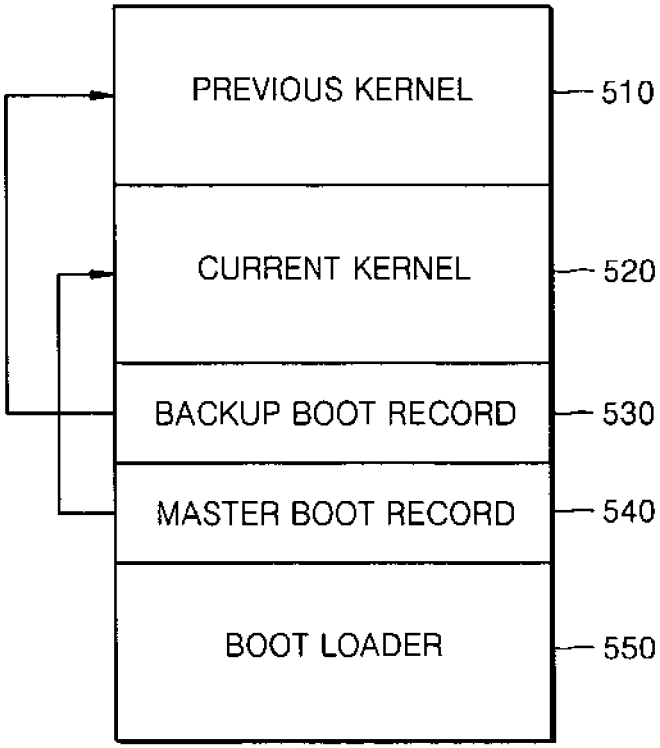

PREVIOUS KERNEL
510
CURRENT KERNEL
520
BACKUP BOOT RECORD
530
MASTER BOOT RECORD
540
BOOT LOADER
550

FIG. 6

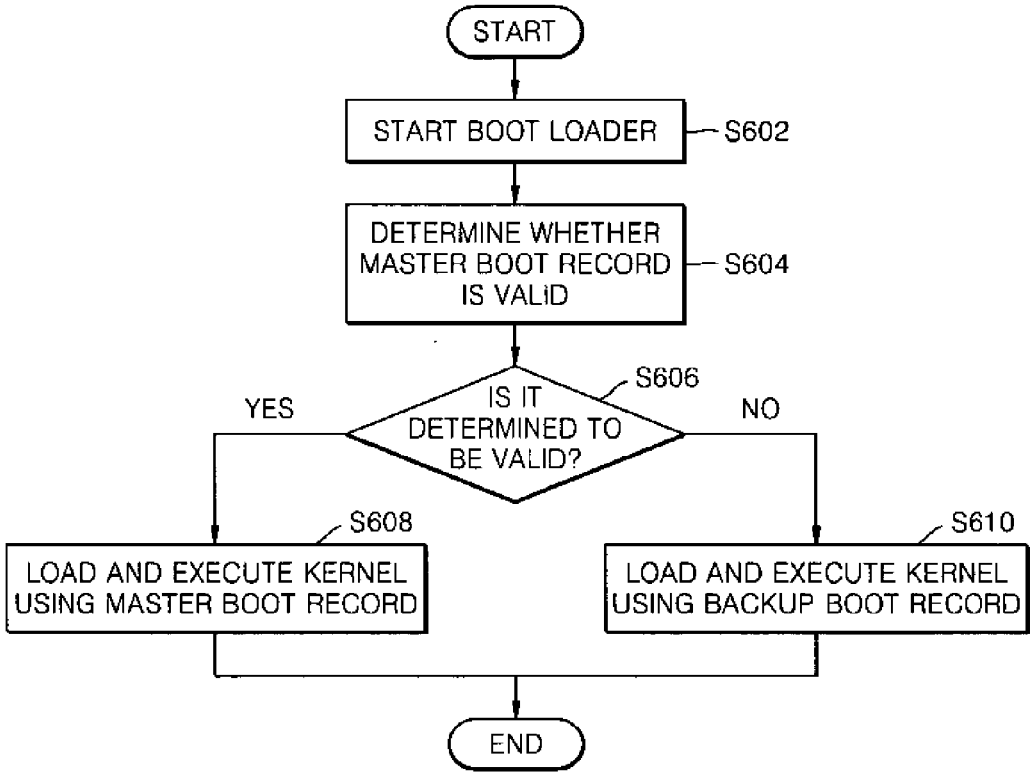

START
START BOOT LOADER
S602
DETERMINE WHETHER MASTER BOOT RECORD IS VALID
S604
IS IT DETERMINED TO BE VALID?
S606
YES
NO
S608
LOAD AND EXECUTE KERNEL USING MASTER BOOT RECORD
S610
LOAD AND EXECUTE KERNEL USING BACKUP BOOT RECORD
END

FIG. 7

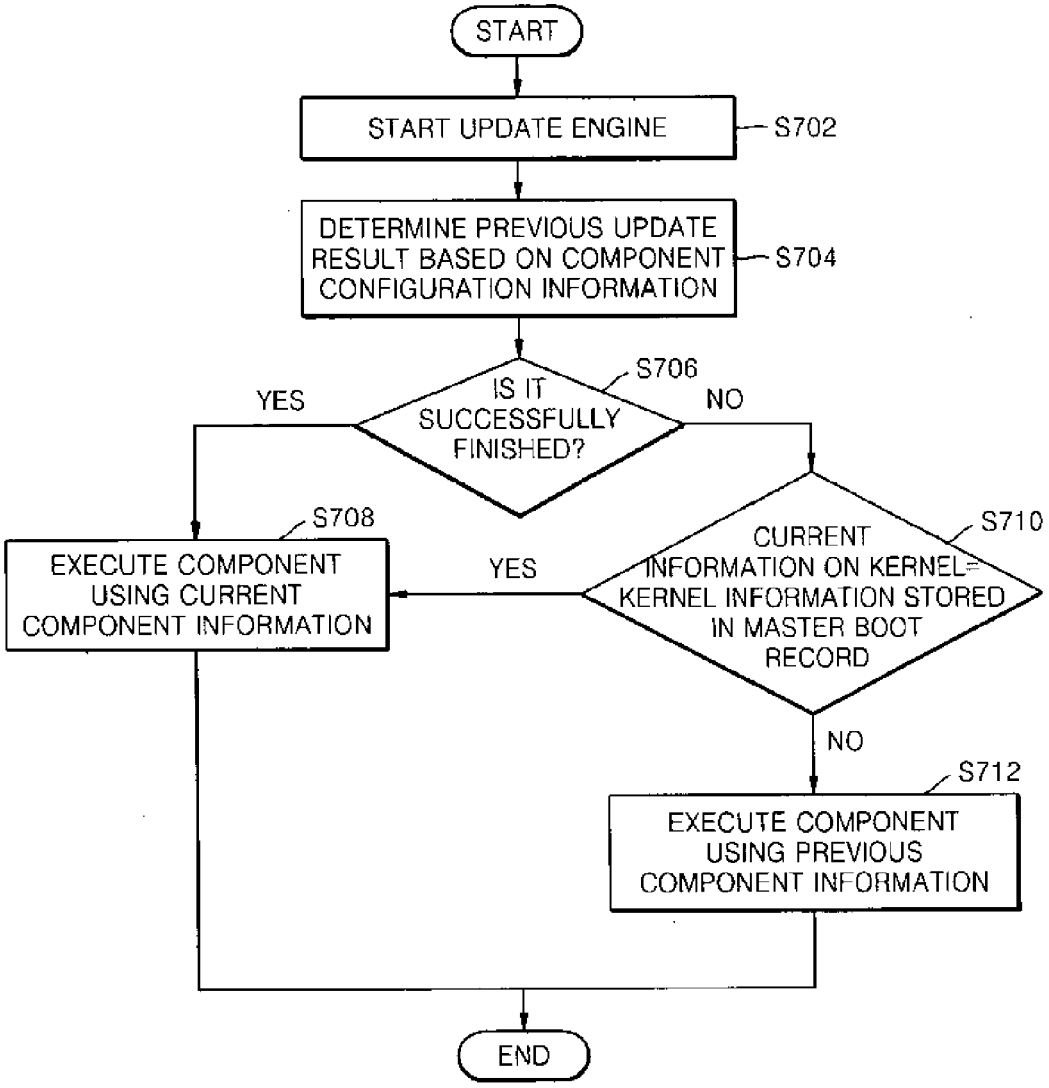

START
START UPDATE ENGINE
S702
DETERMINE PREVIOUS UPDATE RESULT BASED ON COMPONENT CONFIGURATION INFORMATION
S704
IS IT SUCCESSFULLY FINISHED?
S706
YES
NO
EXECUTE COMPONENT USING CURRENT COMPONENT INFORMATION
S708
CURRENT INFORMATION ON KERNEL= KERNEL INFORMATION STORED IN MASTER BOOT RECORD
S710
YES
NO
EXECUTE COMPONENT USING PREVIOUS COMPONENT INFORMATION
S712
END

FIG. 8

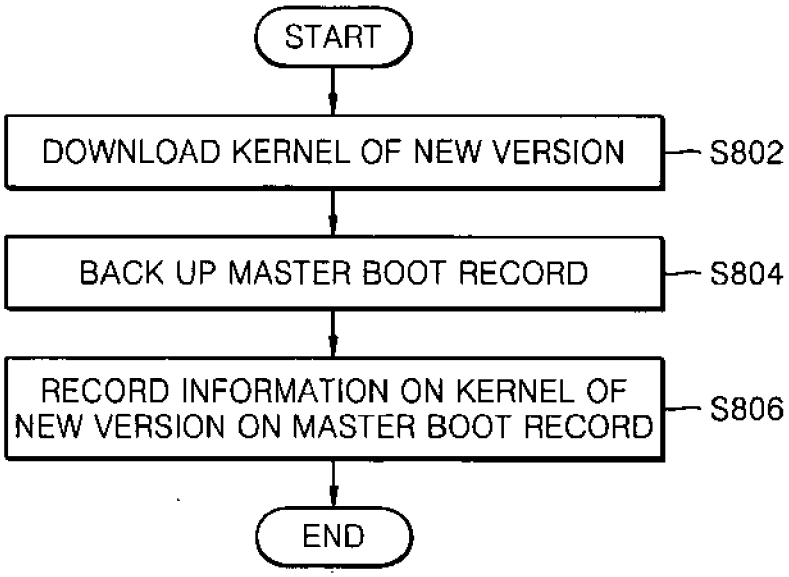
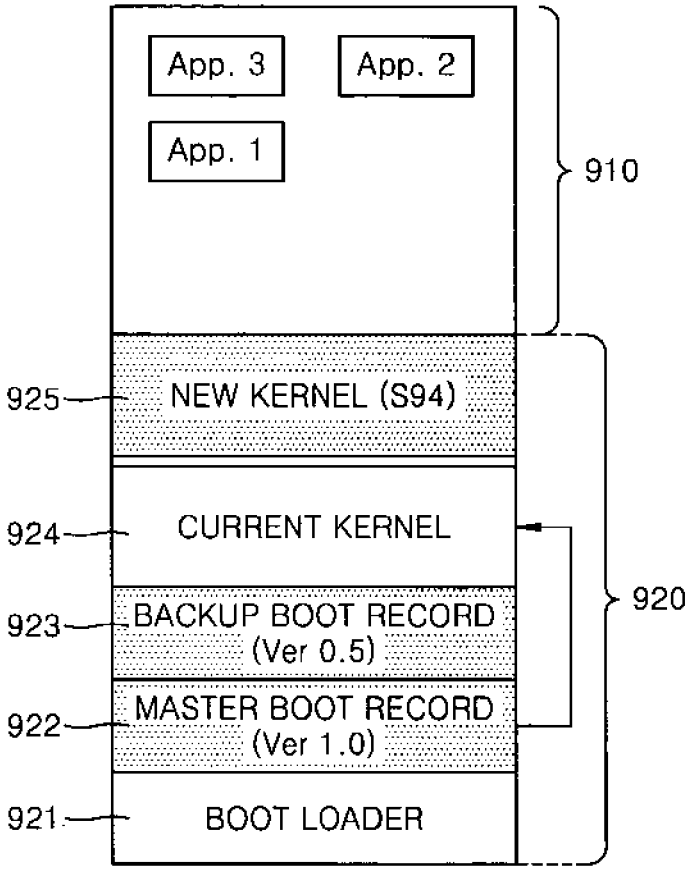

START
DOWNLOAD KERNEL OF NEW VERSION
S802
BACK UP MASTER BOOT RECORD
S804
RECORD INFORMATION ON KERNEL OF NEW VERSION ON MASTER BOOT RECORD
S806
END

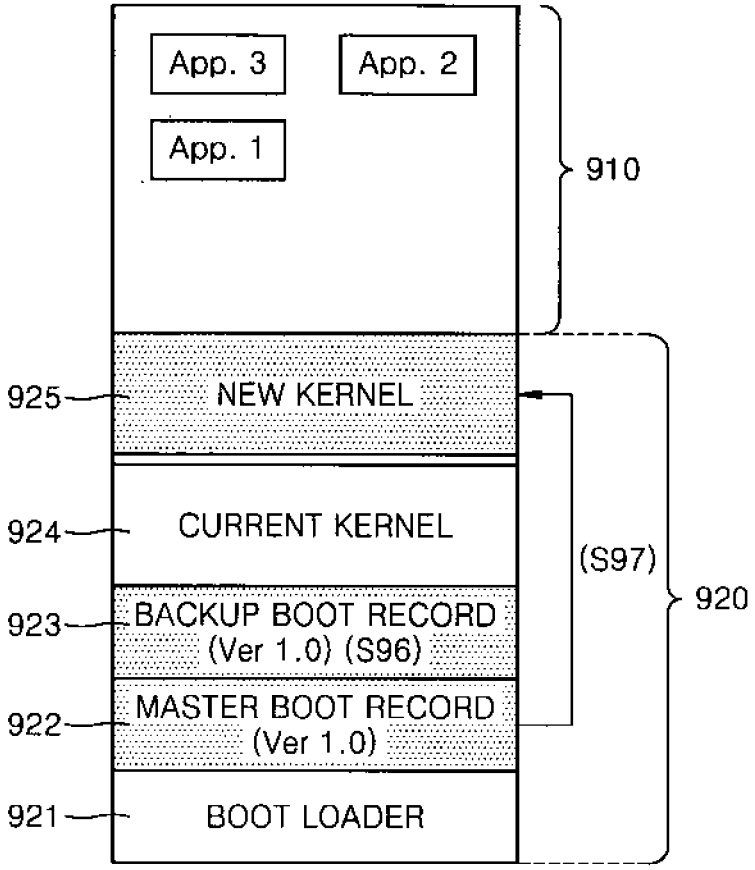
FIG. 9A



CONFIG DB			
PACKAGECOMMIT(940):NO (S91)			
COMP(950)	VER(960)	STATE(970)	PREVIOUS(980)
KERNEL App 1 App 2 App 3	1.0 1.1 2.1 1.1	DOWNLOAD(S92)	1.0(S93)

930

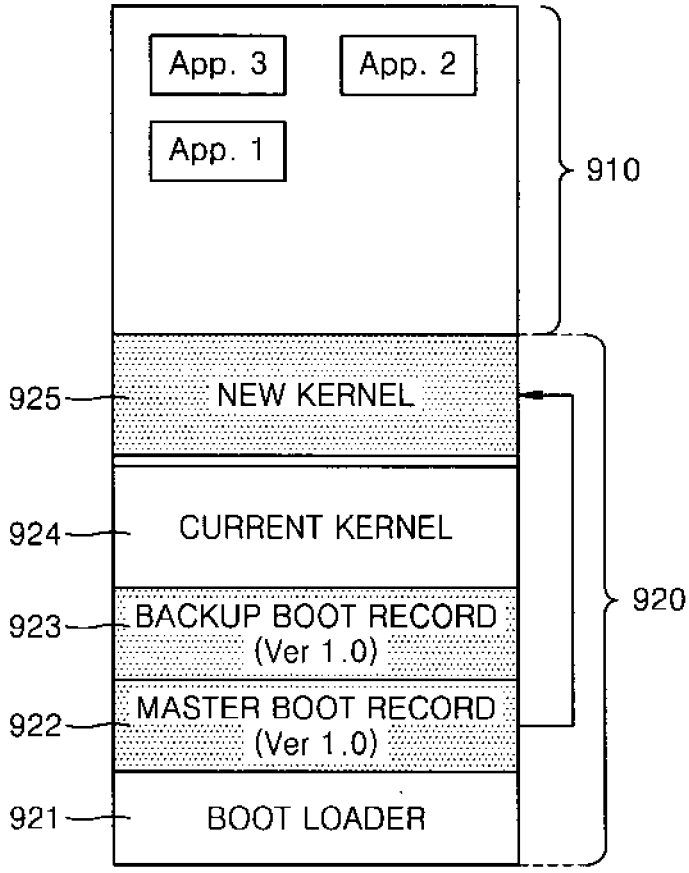
FIG. 9B



CONFIG DB			
PACKAGECOMMIT(940):NO			
COMP(950)	VER(960)	STATE(970)	PREVIOUS(980)
KERNEL App 1 App 2 App 3	1.1(S97) 1.1 2.1 1.1	UPDATE(S95)	1.0

930

FIG. 9C



CONFIG DB			
PACKAGECOMMIT(940):YES(S99)			
COMP(950)	VER(960)	STATE(970)	PREVIOUS(980)
KERNEL App 1 App 2 App 3	1.1 1.1 2.1 1.1	DONE(S98)	1.0

930

FIG. 10A

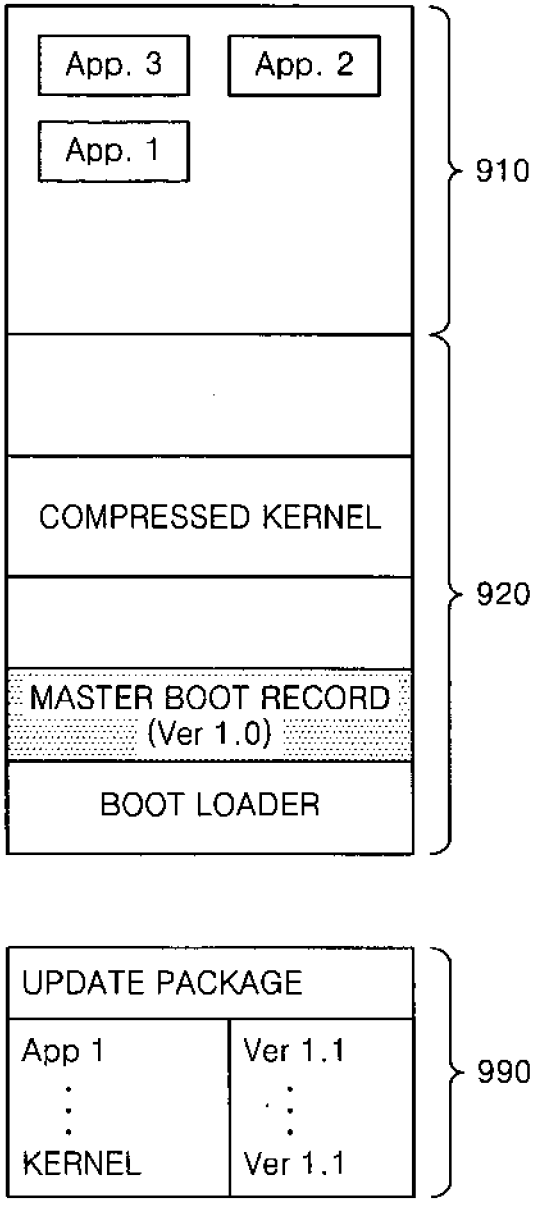
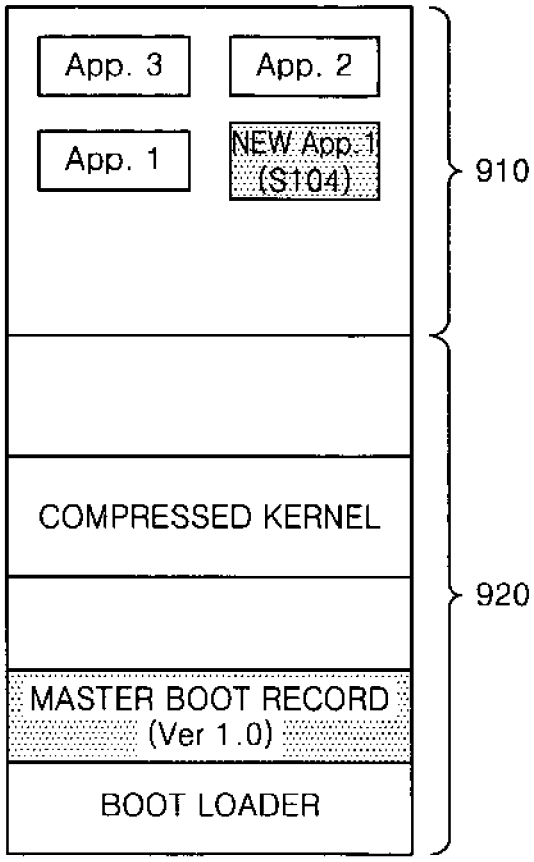

App. 3
App. 2
App. 1
910
COMPRESSED KERNEL
920
MASTER BOOT RECORD
(Ver 1.0)
BOOT LOADER
UPDATE PACKAGE
App 1
KERNEL
Ver 1.1
Ver 1.1
990

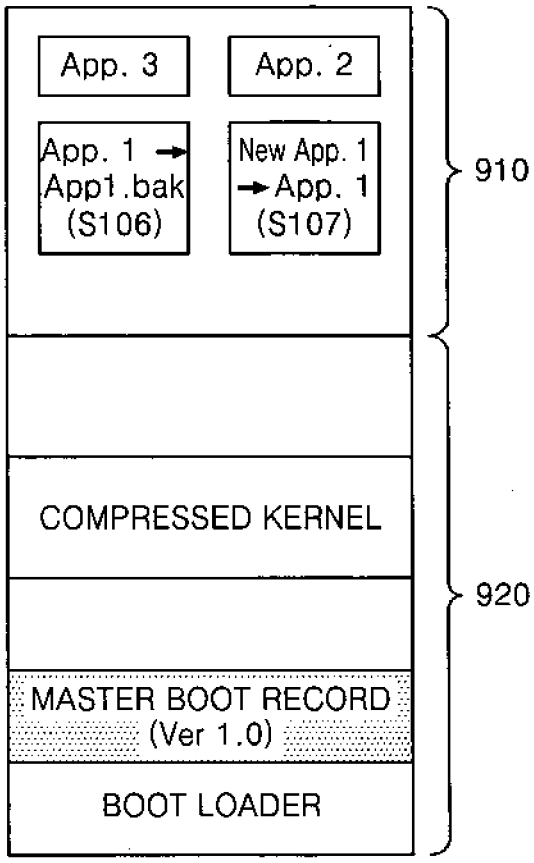
FIG. 10B



CONFIG DB			
PACKAGECOMMIT(940):NO(S101)			
COMP(950)	VER(960)	STATE(970)	PREVIOUS(980)
KERNEL App 1 App 2 App 3	1.0 1.0 2.1 1.1	DOWNLOAD(S102)	1.0(S103)

930

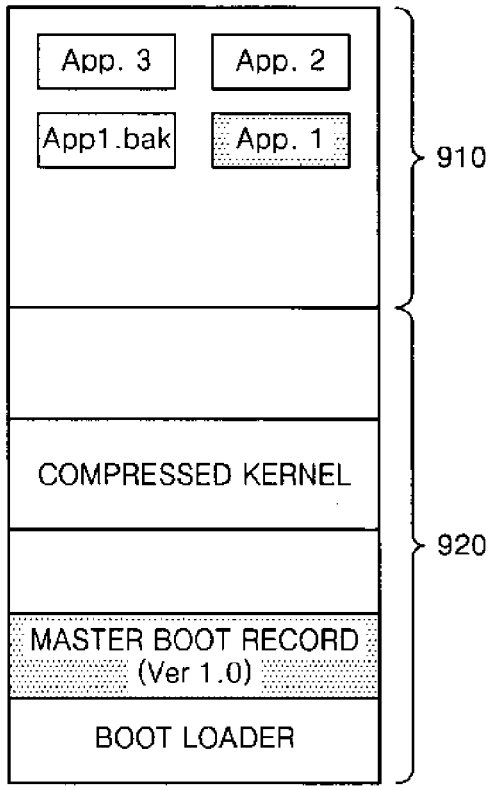
FIG. 10C



CONFIG DB			
PACKAGECOMMIT(940):NO			
COMP(950)	VER(960)	STATE(970)	PREVIOUS(980)
KERNEL App 1 App 2 App 3	1.1 1.0 2.1 1.1	UPDATE(S105)	1.0

930

FIG. 10D



CONFIG DB			
PACKAGECOMMIT(940):YES(S120)			
COMP(950)	VER(960)	STATE(970)	PREVIOUS(980)
KERNEL App 1 App 2 App 3	1.1 1.1(S108) 2.1 1.1	DONE(S109)	1.0

930

FIG. 11

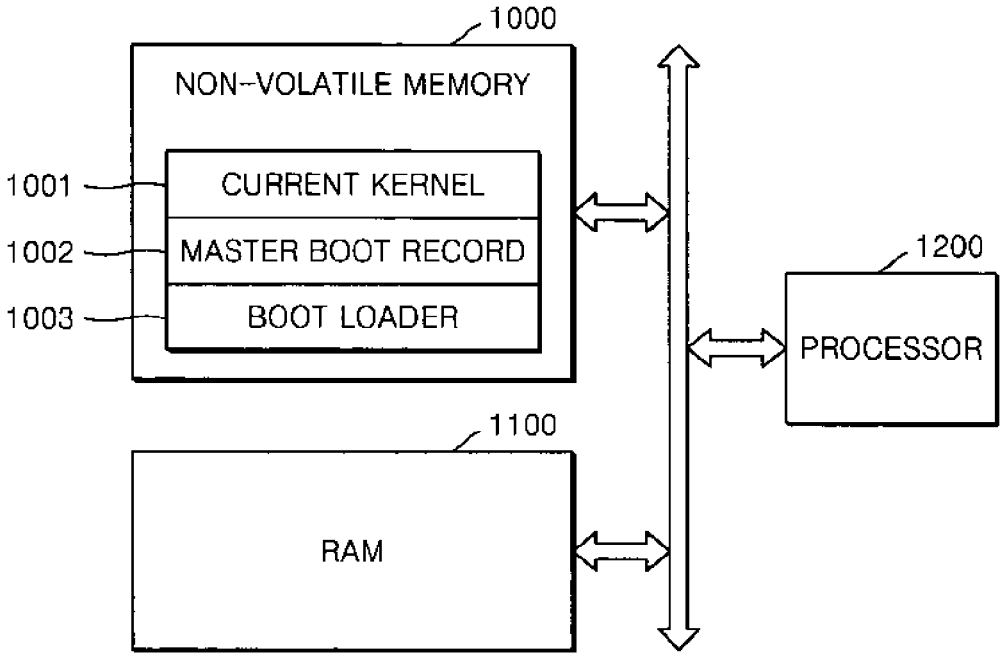
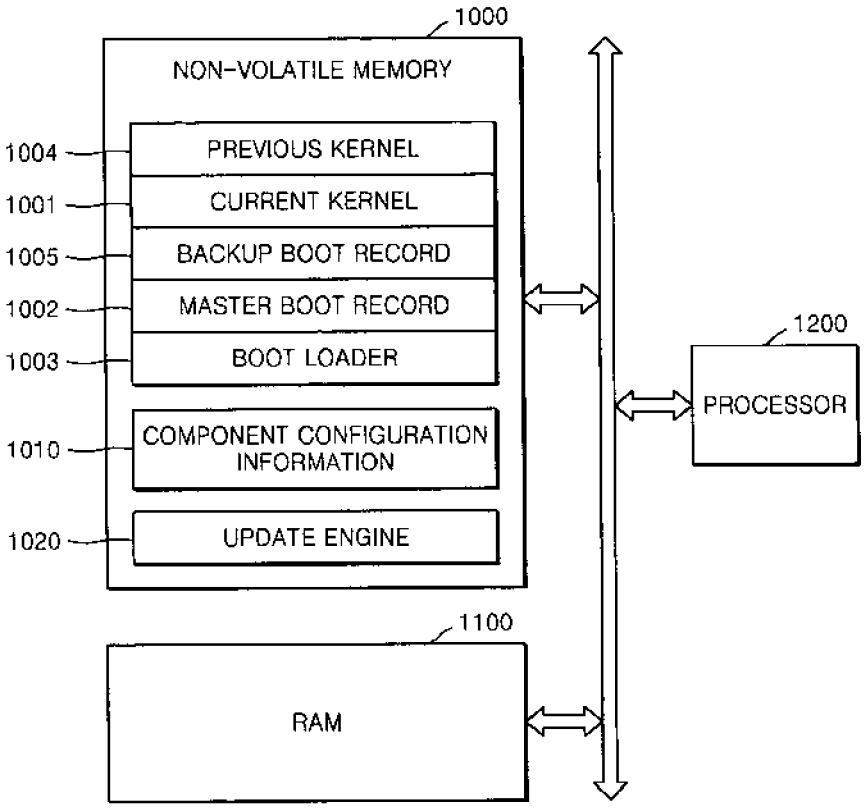

1000
NON-VOLATILE MEMORY
1001
CURRENT KERNEL
1002
MASTER BOOT RECORD
1003
BOOT LOADER
1200
PROCESSOR
1100
RAM

FIG. 12


1000
NON-VOLATILE MEMORY
1004
PREVIOUS KERNEL
1001
CURRENT KERNEL
1005
BACKUP BOOT RECORD
1002
MASTER BOOT RECORD
1003
BOOT LOADER
1010
COMPONENT CONFIGURATION INFORMATION
1020
UPDATE ENGINE
1200
PROCESSOR
1100
RAM

METHOD AND SYSTEM FOR BOOTING AND AUTOMATICALLY UPDATING SOFTWARE, AND RECOVERING FROM UPDATE ERROR, AND COMPUTER READABLE RECORDING MEDIUM STORING METHOD

BACKGROUND OF THE INVENTION

[0001] This is a Continuation of U.S. patent application Ser. No. 13/291,901, filed Nov. 8, 2011, now allowed, which is a Divisional of U.S. patent application Ser. No. 11/513,057, filed Aug. 31, 2006, now U.S. Pat. No. 8,166,285, which claims priority from Korean Patent Application No. 10-2005-0089692, filed on Sep. 27, 2005, in the Korean Intellectual Property Office, the disclosure of which is incorporated herein in its entirety by reference.

[0002] 1. Field of the Invention

[0003] Methods and systems consistent with the present invention relate to updating software, and more particularly, to a booting method for updating software components installed in a system and recovering from an error that occurs during the update, a method and system for automatically updating the software and recovering from the error, and a computer readable recording medium storing the method.

[0004] 2. Description of the Related Art

[0005] A kernel is the core of a related art operating system (OS). When an automatic update of a kernel image is stopped due to a system crash, it is impossible to even boot the OS.

[0006] FIG. **1** is a block diagram of a boot region of a related art system. Referring to FIG. **1**, a current image **110** is a binary image combining an OS and an application. A boot loader **120**, which is a code executed to start a system, loads the current image **110** into random access memory (RAM) and executes the loaded image to boot the system. The boot region has a disadvantage in that when an update of the current image **110** fails, it is impossible to automatically recover the failed update. Also, the boot region does not provide functions of updating several software components and recovering.

[0007] FIG. **2** is a block diagram of a boot region of another related art system, which automatically updates a kernel using an update validity confirmation repository **230**. Referring to FIG. **2**, when the kernel is updated, a current image **220** is copied into a backup repository to generate a backup image **210**, and a new image **220** is downloaded. If there is any trouble with the booting of the system, a boot loader **240** checks an update validity based on the update validity confirmation repository **230** and copies the backup image **210** into the new image **220** to reboot the system. This related art system has a disadvantage in that the entire backup image **210** is copied to recover the system. Therefore, the related art system is not efficient, since it usually takes about 20 to 30 seconds to copy a kernel image of 12 Mbytes (megabytes) in a flash memory.

[0008] FIG. **3** is a block diagram of a boot region of another related art system, which automatically updates a kernel using two boot loaders **330** and **340**. Referring to FIG. **3**, when the kernel is updated, a current image **320** is copied to a backup repository to generate a backup image **310**, and a new image **320** is downloaded. If there is any trouble with the booting when a first boot loader **340** is executed, a second boot loader **330** is executed to load the backup image **310**. This related art system has a disadvantage in that the system should have a function of physically executing two boot loaders **330** and **340**.

[0009] Accordingly, a technology for automatically updating the kernel and effectively recovering the system when an update error occurs is required. Also, the technology must perform an update rollback when an error occurs in updating several software components, including the kernel. More specifically, the technology must automatically recover the software to its previous state (e.g., to a last rebooting point) when the update error occurs, in order to provide reliability of a software automatic update.

SUMMARY OF THE INVENTION

[0010] The present invention provides a booting method of updating software components including a kernel and automatically recovering from an error occurred in the update, a method and system for automatically updating the software and recovering from the error, and a computer readable recording medium storing the method.

[0011] According to another aspect of the present invention, there is provided a method of booting a system by loading a kernel, the method comprising: determining whether a master boot record including information on the kernel is valid; if it is determined that the master boot record is valid, loading and executing the kernel using the master boot record; and if it is determined that the master boot record is not valid, loading and executing a previous kernel using a backup boot record including information on the previous kernel.

[0012] The method may further comprise: determining a previous update result based on component configuration information including information on the update of software components installed in the system; if it is determined that the previous update was successfully finished, loading and executing the components based on current component information included in the component configuration information; and if it is determined that the previous update was stopped, loading and executing the components based on previous component information included in the component configuration information.

[0013] The method may further comprise: when the previous update was stopped, if it is determined that the update of components other than the kernel was completed, and current kernel information included in the component configuration information and kernel information stored in the master boot record are identical to each other, loading and executing the components based on the current component information included in the component configuration information.

[0014] According to another aspect of the present invention, there is provided a method of automatically updating software for updating a kernel and recovering from an update error, the method comprising: downloading the kernel of a new version and storing the downloaded kernel; copying a master boot record including information on the kernel on a backup boot record; and recording information on the kernel of the new version on the master boot record.

[0015] According to another aspect of the present invention, there is provided a method of automatically updating software and recovering from an update error to update a kernel, the method comprising: starting an update transaction; changing state information on the kernel to "download", changing previous information on the kernel to current information, downloading a kernel of a new version, and storing the downloaded kernel; changing state information on the kernel to "update", copying information on a master boot record that includes information on a kernel to a backup boot

record, and recording information on the kernel of the new version in the current information on the kernel and the master boot record; changing the state information on the kernel to "done"; and finishing the update transaction.

[0016] The method may further comprise: before recording new information on the master boot record, indicating a validity flag of the master boot record as "invalid", and, after recording the new information on the master boot record, indicating the validity flag of the master boot record as "valid".

[0017] According to one aspect of the present invention, there is provided a method of automatically updating software for automatically recovering a kernel having an update error and recovering from the update error, the method comprising: determining whether an update of the kernel is stopped; and if it is determined that the update is stopped, copying a backup boot record to a master boot record.

[0018] The determining may further comprise: if a transaction of the update is started, determining that the update is stopped.

[0019] The determining may further comprise: if current information on the kernel is identical to information on the kernel included in the master boot record, determining that the update is successfully finished.

[0020] According to another aspect of the present invention, there is provided a method of automatically updating software for updating more than one software components whose update is necessary and recovering from an update error, the method comprising: sequentially updating the components, other than a kernel, included in an update package including a list of the components whose update is necessary; and if the update package includes the kernel, updating the kernel last.

[0021] The updating of the components may further comprise: changing previous information on the components to current information on the components, and downloading a component of a new version; and changing the current information on the components to information on the component of the new version.

[0022] The method may further comprise: before updating the components other than the kernel, starting an update transaction; and after updating the kernel, finishing the update transaction.

[0023] The updating of the kernel may further comprise: changing state information on the kernel to "download", changing previous information on the kernel to current information, downloading a kernel of a new version, and storing the downloaded kernel; changing state information on the kernel to "update", copying a master boot record that includes information on the current kernel to a backup boot record, and recording information on the kernel of the new version in the current information on the kernel and the master boot record; and changing the state information on the kernel to "done".

[0024] According to another aspect of the present invention, there is provided a method of automatically updating software for recovering software components having an update error and recovering from the update error, the method comprising: determining whether an update of the components included in an update package is stopped; and if it is determined that the update is stopped, converting backup components of the components included in the update package into original components.

[0025] The converting of the backup components may further comprise: if the component is a kernel, copying a backup boot record to a master boot record.

[0026] According to another aspect of the present invention, there is provided a system for automatically updating software and recovering from an update error, the system comprising; a non-volatile memory storing a current kernel, a master boot record including address information on the current kernel, and a boot loader that loads the current kernel to the RAM and executes the RAM using the master boot record to boot the system; and a processor executing the boot loader stored in the non-volatile memory.

[0027] The master boot record may further comprise a validity flag indicating whether the current kernel is valid, the non-volatile memory further comprises a previous kernel and a backup boot record that includes address information on the previous kernel, and if it is determined that the master boot record is valid, the boot loader loads the current kernel using information on the master boot record, and if it is determined that the master boot record is not valid, loading the previous kernel using information on the back boot record.

[0028] The non-volatile memory may further comprise component configuration information that includes information on a software update, and an update engine that updates software components and recovers from an error occurred in the components update using the component configuration information.

[0029] The component configuration information may include state information on an update transaction, previous information including names, addresses, and version of previous components, current information including addresses and version of current components, and update state information on the components, and the update engine updates each of the components and recovers from the update error based on the component configuration information.

BRIEF DESCRIPTION OF THE DRAWINGS

[0030] The above and other aspects of the present invention will become more apparent by describing in detail exemplary embodiments thereof with reference to the attached drawings in which:

[0031] FIG. **1** is a block diagram of a boot region of a related art system;

[0032] FIG. **2** is a block diagram of a boot region of another related art system, which automatically updates a kernel using an update validity confirmation repository;

[0033] FIG. **3** is a block diagram of a boot region of another related art system, which automatically updates a kernel using two boot loaders;

[0034] FIG. **4**A illustrates an example of a boot record according to an exemplary embodiment of the present invention;

[0035] FIG. **4**B is a block diagram of component configuration information according to an exemplary embodiment of the present invention;

[0036] FIG. **5** is a block diagram of a boot region according to an exemplary embodiment of the present invention;

[0037] FIG. **6** is a flowchart illustrating a mechanism of loading a kernel using the boot region illustrated in FIG. **5**;

[0038] FIG. **7** is a flowchart illustrating an operation of loading other software components after the kernel is completely loaded;

[0039] FIG. **8** is a flowchart illustrating a mechanism of automatically updating a kernel according to an exemplary embodiment of the present invention;

[0040] FIGS. **9**A, **9**B, and **9**C in detail illustrate memory state changes during the automatic kernel update according to an exemplary embodiment of the present invention;

[0041] FIGS. **10**A, **10**B, **10**C, and **10**D illustrate memory state changes during the components automatic update of an update package according to an exemplary embodiment of the present invention;

[0042] FIG. **11** is a block diagram of a system for updating software and recovering from an update error according to an exemplary embodiment of the present invention; and

[0043] FIG. **12** is a block diagram of a system for updating software and recovering from an update error according to another exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF THE EXEMPLARY EMBODIMENTS

[0044] Hereinafter, exemplary embodiments of the present invention will be described in detail with reference the accompanying drawings.

[0045] FIG. **4**A illustrates an example of a boot record according to an exemplary embodiment of the present invention. A boot loader loads a kernel and executes the loaded kernel by using the boot record when a system starts.

[0046] The boot record comprises a kernel address **412** and a validity flag **413**. The kernel address **412** indicates where the kernel stored in a flash memory is located. A boot loader finds the kernel using the kernel address **412** and loads the kernel. The validity flag **413** indicates whether the kernel is valid or not. That is, information in the boot record is corrected when the kernel is updated. Before the boot record is corrected, the validity flag **413** is set to indicate that the kernel is invalid, and after the boot record is corrected, the validity flag **413** is set to indicate that the kernel is valid, thereby allowing accurate determination of whether the information in the boot record is valid. Also, a kernel identifier (ID) **410** and a version **411** can be included in the boot record.

[0047] The exemplary embodiment includes two boot records. One of them is a master boot record that stores information on a current kernel, and the other is a backup boot record that includes information on a previous kernel that was successfully loaded.

[0048] FIG. **4**B is a block diagram of component configuration information according to an exemplary embodiment of the present invention. Referring to FIG. **4**B, the component configuration information includes basic information on software components installed in a system and update information. The component configuration information can be realized in a database (DB).

[0049] Transaction state information **420** is used to manage an update transaction and is a flag used to store a current state of the update transaction and determine whether the update succeeds or not.

[0050] The component configuration information contains information on each component **430**. For example, but not by way of limitation, the information comprises a name **431**, current component information **440** including address information **432** and version information **433**, previous component information **450** for managing a previous component address **434** and a previous version **435** before the current component is updated, and component state information **436** for managing a state of the component update. The exemplary embodiment automatically recovers the system using the component state information **436**.

[0051] FIG. **5** is a block diagram of a boot region according to an exemplary embodiment of the present invention. Referring to FIG. **5**, the boot region comprises a boot loader **550**, a current kernel **520**, a backup previous kernel **510**, a master boot record **540** in which information on the current kernel **520** is recorded, and a backup boot record **530** in which information on the previous kernel **510** is recorded.

[0052] FIG. **6** is a flowchart illustrating a mechanism of loading a kernel using the boot region illustrated in FIG. **5**. Referring to FIG. **6**, the boot loader **550** is started (Operation S**602**) to determine whether a validity flag of the master boot record **540** is valid or not (Operation S**604**). If it is determined that the validity flag of the master boot record **540** is valid (Operation S**606**), the kernel is loaded and executed using the master boot record **540** (Operation S**608**). That is, the current kernel **520** indicated by the master boot record **540** is loaded. However, if it is determined that the validity flag of the master boot record **540** is not valid (Operation S**606**), which indicates an error occurs during a previous update of kernel, the previous kernel **510** is loaded and executed using the backup boot record **530** (Operation S**610**) to stably start the system.

[0053] FIG. **7** is a flowchart illustrating an operation of loading other software components after the kernel is completely loaded. Referring to FIG. **7**, an update engine program loads and executes the software components used in a system. An update engine is started (Operation S**702**) to determine whether a previous update is successfully finished based on component configuration information (Operation S**704**). The determination is based on transaction state information of the component configuration information. If the transaction state information indicates that a transaction has ended, an update transaction is characterized as successfully finished. If the transaction state information indicates that the transaction has started, the update transaction is characterized as unsuccessfully stopped.

[0054] If it is determined that the previous update is successfully finished (Operation S**706**), the software components are loaded and executed using current component information included in the component configuration information (Operation S**708**). That is, the current versions of software components are loaded based on current component address information.

[0055] However, if it is determined that the previous update is stopped (Operation S**706**), the software components are loaded and executed using previous component information included in the component configuration information (Operation S**712**). That is, the software components of a previous version are loaded based on previous component address information.

[0056] According to an exemplary embodiment of the present invention, the kernel may be updated after all applications other than the kernel are updated. Therefore, although the previous update is stopped, it may be determined that an update of the software components other than the kernel is successfully finished, and current kernel information included in the component configuration information is the same as kernel information of the master boot record (Operation S**710**). This means that the kernel update is stopped after information on the master boot record is successfully corrected. In this case, since it may be determined that an update transaction is successfully finished even if the transaction has not been committed, it is efficient to load and execute the

software components using the current component information included in the component configuration information (Operation S**708**).

[0057] Even if the update of several components is stopped before the system is rebooted, the system can be recovered to a configuration of components last used by a user.

[0058] FIG. **8** is a flowchart illustrating a mechanism of automatically updating a kernel according to an exemplary embodiment of the present invention. Referring to FIG. **8**, the kernel is updated by downloading the kernel of a new version (Operation S**802**), backing up a master boot record including information on the kernel in a backup boot record (Operation S**804**), and recording information (e.g., address, etc., but not limited thereto) on the kernel of the new version to the master boot record (Operation S**806**).

[0059] FIGS. **9**A, **9**B, and **9**C illustrate in detail memory state changes during the automatic kernel update according to an exemplary embodiment of the present invention. Referring to FIGS. **9**A, **9**B, and **9**C, a file system region **910** stores three applications App. 1, App. 2, and App. 3, and a raw format region **920** has the substantially same constitution as the boot region illustrated in FIG. **5**. A configuration DB **930** that stores component configuration information necessary for the update and recovery contains PACKAGECOMMIT **940**, which is transaction state information, component identifiers **950**, current version information **960**, component state information **970**, and previous version information **980**. Although not shown in FIGS. **9**A, **9**B, and **9**C, the current address information **423** and the previous address information **435** illustrated in FIG. **4**B are updated when the version information is updated.

[0060] The raw format region **920**, which is a boot area excluding a file system, stores a boot loader **921**. However, if the boot loader **921** includes the file system, a kernel and a boot record may be stored in the file system region **910**. Also, since the file system region **910** stores the file system like a disk operation system (DOS), the exemplary embodiment does not require such a file system.

[0061] If the exemplary embodiment does not have a file system, applications can be stored in the raw format region **920**. Also, the configuration DB **930** may be stored in one of the raw format region **920** and the file system region **910**.

[0062] Referring to FIG. **9**A, PACKAGECOMMIT **940** is set as "NO" (Operation S**91**), which means that the update transaction is started and not yet committed. The state information **970** of the kernel to be updated is changed to "DOWNLOAD" (S**92**), previous version information **980** is changed to "1.0" corresponding to the current version information **960** (S**93**), and a new kernel **925** of a new version (**1.1**) is downloaded and stored (S**94**). The master boot record **922** indicates a current version **1.0** kernel **924**, whereas the backup boot record **923** includes information on a previous version **0.5** backup kernel.

[0063] Referring to FIG. **9**B, kernel state information is changed to "UPDATE" (S**95**), information on the master boot record **922** is copied to the backup boot record **923**, and information on the kernel of the new version is recorded in current kernel information and the master boot record **922**. More specifically, the master boot record **922** indicates a newly downloaded version 1.1 kernel **925** (S**97**), and the corresponding current version and address of the configuration DB **930** are accordingly changed.

[0064] Referring to FIG. **9**C, the kernel state information is changed to "DONE" (S**98**) to indicate that the kernel is successfully updated. Finally, PACKAGECOMMIT **940** is set as "YES" (S**99**) to indicate that the update transaction has been successfully finished.

[0065] In the above process, the validity flag of the master boot record **922** is always indicated "invalid" before new information is recorded on the master boot record **922**, and the validity flag of the master boot record **922** is always indicated "valid" after new information is recorded on the master boot record **922**.

[0066] When an error occurs in the kernel update, the system can be automatically recovered to the kernel of a successfully loaded version by determining whether the kernel update is successfully finished, and, if it is determined that the kernel update is stopped, copying the backup boot record **923** to the master boot record **922**.

[0067] Whether the kernel update is stopped is determined based on PACKAGECOMMIT **940**. In detail, when the system is booted, if PACKAGECOMMIT **940** is set as "NO", it is determined that the update transaction has been stopped.

[0068] However, even if the update transaction fails, the current kernel information and the kernel information stored in the master boot record **922** may be substantially identical to each other, that is, when the update transaction is stopped after S**91** through S**97** are performed. In this case, it is regarded that the kernel update is successfully finished, and the update transaction may be committed to finish without the automatic recovery process.

[0069] FIGS. **10**A, **10**B, **10**C, and **10**D illustrate memory state changes during automatically updating the components in an update package **990** according to an exemplary embodiment of the present invention.

[0070] Referring to FIG. **10**A, the update package **990** is a list of software components to be updated. The list may include application components and a kernel. It may be possible to sequentially update the application components (S**102** through S**109**) and then update the kernel that is a critical component.

[0071] An application component other than the kernel, i.e., an application App. 1 included in the update package **990**, is updated as illustrated in FIGS. **10**B, **10**C, and **10**D. The application App. 1 is updated by changing previous component information to current component information, downloading components of a new version, and changing the current component information to information on the new version component.

[0072] Referring to FIG. **10**B, PACKAGECOMMIT **940** is set as "NO" to start an update transaction (Operation S**101**). The current component state information **970** is changed to "DOWNLOAD" (Operation S **102**), the previous component information **980** is changed to the current component information **960** (Operation S**103**), and a component, i.e., a new APP. 1, of a new version is downloaded and stored (Operation S **104**).

[0073] Referring to FIG. **10**C, the component state information **970** is changed to "UPDATE" (Operation S**105**), the component APP. 1 of a previous version is backed up as App 1.bak (Operation S**106**), and the component, i.e., the new APP. 1, of the new version is changed to the original component App. 1 (Operation S**107**).

[0074] Referring to FIG. **10**D, the current component information **960** is changed to information on the component of the new version (Operation S**108**), and the component state infor-

mation **970** is changed to "DONE" (Operation S**109**). The current and previous versions of the component App.1 are 1.1 and 1.0, respectively.

[0075] Each of the applications included in the update package **990** is updated through Operations S**102** through S **109**. The kernel is finally updated through Operations S**92** through S**98** illustrated in FIGS. **9**A, **9**B, and **9**C. When every component included in the update package **990** is completely updated, PACKAGECOMMIT **940** is set as "YES" to finish the update transaction (Operation S**120**).

[0076] When an error occurs in the component update, the components can be automatically recovered to the previous version by determining whether components included in the update package are successfully updated, and, if it is determined that the component update is stopped, changing backup components corresponding to all the components included in the update package to original components. When an error occurs in the kernel update, the kernel can be automatically recovered to the previous version by copying the backup boot record **923** to the master boot record **922**.

[0077] Whether the update error occurs is determined based on state information on the update transaction. If the update transaction is in a state that indicates a start when the system is rebooted, it is determined the update transaction was unsuccessfully stopped before being rebooted. However, although the update transaction is not finished, if the state information on the components included in the update package **990** other than the kernel is set as "DONE", and the current kernel information and the kernel information stored in the master boot record **922** are identical to each other, the kernel update is actually finished. Therefore, it is regarded that the kernel update is successfully finished, the kernel may not be recovered to the previous version and the update transaction is committed.

[0078] FIG. **11** is a block diagram of a system for updating software and recovering from an update error according to an exemplary embodiment of the present invention. Referring to FIG. **11**, the system comprises a processor **1200**, a RAM **1100**, and a non-volatile memory **1000** that stores a current kernel **1001**, a master boot record **1002** including address information on the current kernel **1001**, and a boot loader **1003** that is a program code (e.g., a set of instructions) for booting the system.

[0079] The boot loader **1003** loads the current kernel **1001** to the RAM **1100** and executes the RAM **1100** using the master boot record **1002** to boot the system. The master boot record **1002** may include a validity flag indicating whether the current kernel **1001** is valid.

[0080] FIG. **12** is a block diagram of a system for updating software and recovering from an update error according to another exemplary embodiment of the present invention. The non-volatile memory **1000** may include a previous kernel **1004** of a previous version that is a backup kernel, and a backup boot record **1005** that is a backup master boot record **1002**. In this case, when the system is booted, the boot loader **1003** determines whether the master boot record **1002** is valid. If it is determined that the master boot record **1002** is valid, the current kernel **1001** is loaded using information on the master boot record **1002**, and if it is determined that the master boot record **1002** is not valid, the previous kernel **1004** is loaded using information on the backup boot record **1005**.

[0081] The non-volatile memory **1000** may include component configuration information **1010** including information on a software update, and an update engine **1020** that updates software components and recovers from an error that occurs in the components update using the component configuration information **1010**.

[0082] The component configuration information **1010** may include state information on an update transaction, previous information including names, addresses, and version of components which were installed before they are updated, current information including addresses and version of updated components, and update state information on the components, as illustrated in FIG. **4**B. The update engine **1020** updates each of the components and recovers from the update error based on the component configuration information **1010**.

[0083] The exemplary embodiments can also be embodied as computer readable code (e.g., a set of instructions) on a computer readable medium.

[0084] As described above, the exemplary embodiment can update software and a kernel that may cause a fatal result when its' update fails at that time, and can recover a system from an update error using component configuration information.

[0085] Also, even if the software update is stopped for an external reason, the exemplary embodiment recovers a system in a booting unit using final configuration information that is successfully loaded, thereby substantially perfectly rolling back the system to a previous version.

[0086] Also, the exemplary embodiment effectively update kernel and boot the system using boot records, and substantially improves a system operating speed by changing only the boot records when the kernel is recovered.

[0087] While this invention has been particularly shown and described with reference to exemplary, non-limiting embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

What is claimed is:

1. A method of updating software, the method comprising:

updating a plurality of software components listed by an update package;

determining whether the updating the software components has failed; and

in response to determining that the updating the software components has failed, recovering from an update error by converting backup components of the software components into corresponding original software components.

2. The method of claim **1**, wherein the updating the software components comprises:

copying a master boot record including information on a kernel of a current version on a backup boot record; and

updating the kernel of the current version into a kernel of a new version by recording information on the kernel of the new version on the master boot record.

3. The method of claim **2**, wherein the recovering comprises:

determining whether the updating the kernel of the current version into the kernel of the new version has failed; and

in response to determining that the updating the kernel of the current version into the kernel of the new version has failed, copying the backup boot record to the master boot record.

4. The method of claim **1**, wherein the determining comprises:

checking state information of a transaction for the updating the software components; and

in response to the checked state information indicating that the transaction is done, determining whether the updating the software components has succeeded.

5. A non-transitory computer readable medium having embodied thereon a computer program for executing a method of updating software, the method comprising:

updating a plurality of software components listed by an update package;

determining whether the updating the software components has failed; and

in response to determining that the updating the software components has failed, recovering from an update error by converting backup components of the software components into corresponding original software components.

6. An apparatus configured to update software, the apparatus comprising:

a memory configured to store information on a software update and an update engine for updating software using the information on the software update; and

a processor configured to execute the update engine,

wherein the update engine is configured to update a plurality of software components listed by an update package, determine whether the update of the software components has failed, and in response to determining that the update of the software components has failed, recover from an update error by converting backup components of the software components into corresponding original software components.

7. The apparatus of claim **6**, wherein the memory is configured to store a kernel of a current version, a master boot record including information on a kernel of a current version, and a backup boot record, and

the update engine is configured to copy the master boot record on the backup boot record and update the kernel of the current version into a kernel of a new version by recording information on the kernel of the new version on the master boot record.

8. The apparatus of claim **7**, wherein the update engine is configured to determine whether the update of the kernel has failed, and, in response to determining that the update of the kernel has failed, copy the backup boot record to the master boot record.

9. The apparatus of claim **6**, wherein the update engine is configured to check state information of a transaction for the update, and, in response to the checked state information indicating that the transaction is done, determine whether the update of the software components has succeeded.

* * * * *