(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 February 2017 (02.02.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/019086 A1

(51) International Patent Classification:
G06F 9/455 (2006.01)

(21) International Application Number:
PCT/US2015/042891

(22) International Filing Date:
30 July 2015 (30.07.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **STEIN, Daniel**; 4245 Technology Drive, Fremont, California 94538 (US). **NAZARI, Siamak**; 4245 Technology Drive, Freemont, California 94538 (US).

(74) Agents: **ORTEGA, Arthur S.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

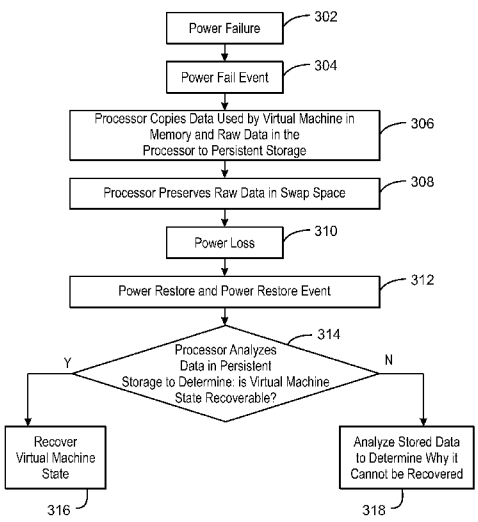
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

(54) Title: PRESERVING VIRTUAL MACHINE DATA

(57) Abstract: Example implementations relate to preserving virtual machine data during a power fail event. For example, a system can include a processor, a memory, and a storage including a swap space used by the virtual machine. The system may include a power element to deliver power to the system during a power fail event. In response to the power event, the processor can copy data located in the processor and memory to a persistent storage and also preserve the data located in the swap space.



300
FIG. 3

WO 2017/019086 A1

PRESERVING VIRTUAL MACHINE DATA

BACKGROUND

[0001] A computing device can host emulations or virtualizations of other computing systems on its hardware. Virtualizing another computing system on a first computing device allows the use of virtualized machine software or architecture while avoiding implementation of the machine software or architecture on a second computing device. The virtualized machine uses the hardware components of the computing device upon which it is implemented.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Certain examples are described in the following detailed description and in reference to the drawings, in which:

[0003] FIG. 1 is an example of a computing system for saving a virtual machine during a power fail event;

[0004] FIG. 2 is an example of a conceptual layout of a virtual machine across hardware components of a host computing device;

[0005] FIG. 3 is a block diagram of an example method for saving and restoring a virtual machine state;

[0006] FIG. 4 is block diagram of an example method for saving a virtual machine during a power fail event; and

[0007] Fig. 5 is a diagram of a an example non-transitory, computer-readable medium that holds code that when executed by a processor saves a virtual machine state on in persistent storage.

DETAILED DESCRIPTION OF SPECIFIC EXAMPLES

[0008] A virtual machine kernel can be a virtualization infrastructure for use on an operating system that can turn a virtual machine into a hypervisor for presenting a guest operating system with a virtual operating platform for management of the execution of the guest operating systems. A virtual machine can be an emulation of a computer system. Virtual machines can operate based on the computer architecture and functions of a real or hypothetical computer, and the implementations of a virtual machine may involve specialized hardware, software, or a combination of both.

[0009] Present virtual machines (VM) have a mechanism for stopping, saving, and restarting the VM from a saved image file. This process does not account for time constrained nature of a computing device experiencing a loss of power.

[0010] Methods and techniques described herein can apply to VMs and can also be generalized to anything that needs to survive a power fail event. The hardware of a machine can maintain power for a limited time, and during this time the state of a machine can be preserved. The amount of time for preserving the state of the machine can be bounded by the amount of physical state that needs to be preserved.

[0011] Many applications have a transactional model to support restarting. An application restarts the transactional state of a device, when the device itself can be restarted after a power fail event. One example of preserving a state of a transactional model includes preserving a log of unacknowledged requests. The present disclosure preserves metadata for restoring the application before power and any back up power supply that allows the machine to function, is disabled.

[0012] In an example, the metadata preserved can be found in anonymous memory, such as the swap space, and also in physical memory such as random access memory (RAM). In the present disclosure, before the physical memory is saved in persistent storage, all hardware caches can be flushed to physical memory. In the present disclosure, a filter is used to preserve a subset of the metadata that can be limited to exclude any metadata not related to restart. Use of this filter can further bound the time it takes to save metadata as it can limit the metadata to be saved to the physical limits of memory and anonymous memory or swap space.

[0013] In an example, the metadata to be saved can be the raw data for reconstructing the saved state for use in restarting the application from a power fail event checkpoint. The data can be raw in that it is unformatted, and accordingly, during a power fail event, takes no extra time for formatting during preserving and copying. In an example for VMs, the metadata that was preserved can be reconstituted when the machine reboots. Reconstitution can consist of a machine checkpoint and the machine restore combined. In the present disclosure, and examples that preserves limited subsets of metadata, the format of how the data is to be restored can be embedded in the logic of power fail restore. This format can be optimized for VMs or generalized to work with any application.

[0014] Methods disclosed herein ensure the process of saving a virtual machine state can be accomplished in a bounded amount of time. Bounding the amount of time for saving a virtual machine state can improve the resilience of a device, which runs virtual machines, to losses of power. In one example, the recovery of a virtual machine state can include first storing backup of a virtual machine and its linked files without regard for how or where the virtual machine state is stored. The present disclosure includes copying virtual machine state data from volatile memory. In an example, the data on all volatile memory devices is saved. By limiting the amount and location of the saved data by explicitly saving 'all volatile memory,' the present disclosure bounds the time of saving because the volatile memory of each device has unique and limited physical limits of volatile memory installed on the host device. In another example, the swap space used by the virtual machine and located on a physical storage is preserved even during power fail event. A swap space, also known as a swap file or page file, can be designated space on a hard disk used to be used as a virtual memory extension of a computer's real memory like random access memory (RAM). Having a swap file allows your computer's operating system to use the small amount of hard disk space as an extension of the system memory and data stored in a swap space can swap from the swap space to the memory and processor as needed for processing or quicker access.

[0015] A power fail event can be a point in time during which a device losses power, and the power loss event can persist while a loss of power to the device can be ongoing, however a power fail event begins when power is first lost or insufficiently provided to meet a device's power consumption needs. Thus a power fail event can signal for a data preserving process to begin.

[0016] This preserved swap space can also be limited in size and can be used for restoration of the virtual machine upon power restore. Like the hardware memory constraints, swap space is similarly bounded by partitions in hardware such as a hard drive. In an example, the amount of data to save or preserve is kept manageable by limiting the data to be saved to include the data in use by the virtual machine. This limited data can include the virtual machine kernel, data stored in the caches of the central processing unit, swap space of a storage, or user data in a memory device, such as memory implementing random access memory (RAM).

[0017] FIG. 1 is an example of a computing system for saving a virtual machine during a power fail event. The computing device 100 may be, for example, a laptop computer, desktop computer, ultrabook, tablet computer, mobile device, or server, among others. The computing device 100 may include a central processing unit (CPU) 102 that is configured to execute stored instructions, as well as a memory device 104 that stores instructions that are executable by the CPU 102. The CPU 102 may be coupled to the memory device 104 by a bus 106. The CPU 102 can be a single core processor, a multi-core processor, a computing cluster, or any number of other configurations. Furthermore, the computing device 100 may include more than one CPU 102. The CPU 102 can also connect through a storage array interface 106 to external storage arrays 110 by the bus 106. The storage array 110 can be an external system or array of systems that are hosting its own guest virtual machines or interacting with the virtual machines of the computing device 100.

[0018] The computing device 100 also can locally include a storage device 112. The storage device 112 is a non-volatile storage device such as a hard drive, an optical drive, a thumbdrive, an array of drives, or any combinations thereof. The memory device 104 can include random access memory (RAM), read only memory (ROM), flash memory, or any other suitable memory systems. For example, the memory device 104 may include dynamic random access memory (DRAM).

[0019] The CPU 102 may be linked through the bus 106 to a display interface configured to connect the computing device 100 to a display device. The display device may include a display screen that is a built-in component of the computing device 100. The display device may also include a computer monitor, television, or projector, among others, that is externally connected to the computing device 100 or built into the computing device 100, for example, in a laptop or tablet.

[0020] The CPU 102 may also be connected through the bus 106 to an input/output (I/O) device interface configured to connect the computing device 100 to one or more I/O devices. The I/O devices may include, for example, a keyboard and a pointing device, wherein the pointing device may include a touchpad or a touchscreen, among others. The I/O devices may be built-in components of the computing device 100 or may be devices that are externally connected to the computing device 100.

[0021] The computing device 100 may also include a network interface controller (NIC) may be configured to connect the computing device 100 through the bus 106 to the network. The computing device 100 may also be connected to a network. The network may be a wide area network (WAN), local area network (LAN), or the Internet, among others.

[0022] The computing device 100 and its components may be powered by a power supply unit (PSU) 114. The CPU 102 may be coupled to the PSU through the bus 106 which may communicate control signals or status signals between then CPU 102 and the PSU 114. The PSU 114 is further coupled to a power source 116. The power source 116 can be a supply external to the computing device 100, and can also be internal in the case of a battery, or can be both in the case of an external power source with a power supply backup to continue supplying power and send a power fail event in the case of a power fail. In examples with a power source 116 that includes a power source backup, the CPU 102 can control the functioning of the backup over a bus 106.

[0023] The computer device 100 also includes a virtual machine state recoverer (VMSR) 118, which may be stored in the storage device 112. As disclosed herein, the VMSR 118 may instruct the processor to copy volatile data of a virtual machine in a memory device 104 or stored in the memory stores of the CPU 102. The VMSR 118 can also instruct the preservation of data used by a virtual machine in the swap space of a storage device 112.

[0024] Further, the VMSR 118 can aid in the recovery of a virtual machine state upon return of power to a computing device 100. In an example, the VMSR 118 can direct a processor to load and restore a virtual machine state by indicating the location in a persistent storage of the saved virtual machine state data. Persistent storage is non-volatile such that data stored there is preserved even without power being supplied to the persistent storage. Further, persistent storage can allow the storage of a state of an application or process through serialization of the data to a storable format, and then saving this data to a file for future retrieval. In an example, the virtual machine state is stored in a local version of persistent storage such as a storage device 112. The virtual machine state can also be stored remotely at a storage array 110, for example.

[0025] The block diagram of Fig. 1 is not intended to indicate that the computing device 100 is to include all of the components shown in Fig. 1. Further, the computing device 100 may include any number of additional components not shown in Fig. 1, depending on the details of the specific implementation.

[0026] FIG. 2 is an example of a conceptual layout 200 of a virtual machine across hardware components of a host computing device. Like numbered items are as described in Fig. 1. For example, the host computing device can be the computing device 100 described in Fig. 1 and can host a guest virtual machine, or several machines such as the virtual machine 202 shown.

[0027] The virtual machine 202 can be implemented across one or several memory, processing, and storage devices as shown. Indeed, more than one virtual machine 202 can be hosted on a computing device 100, and further, a virtual machine 202 can be hosted across several different computing devices making use of devices resources. For simplicity, a computing device 100 and virtual machine 202 are shown.

[0028] As the virtual machine 202 is emulated on the hardware of the computing device 100, it can take up either a portion of those resources, or may completely control of these components functioning. As an illustration of the potential partial resources usage of a virtual machine 202 on a computing device 100, a part of the computing device 100 resources is shown inside the virtual machine 202. In one example, the present drawings do not represent the relative amount of control of the virtual machine 202. The present drawings provide an example of the relationship of the indicated components with the virtual machine 202 and the computing device 100.

[0029] The computing device 100 can include a CPU 102, a memory device 104, and a storage device 112. These items are as described above. A virtual machine hosted on the computing device 100 can make use of these components. The CPU caches 204 are potential locations for storage and use by a virtual machine 202. The CPU caches can include the L1, L2, and L3 caches and are decreasingly volatile and fast. The CPU caches can also host data used in the virtual machine 202. Upon receipt of a power fail event, the CPU 102 can be instructed to copy data in its or other CPU caches 204 to be copied to persistent storage, and thereby save a part of a virtual machine 202 state. In an example, the virtual machine 202 state

data is stored in the CPU caches 204 and no other copying or preserving of data is undertaken.

[0030] In another example, the storage device 112 can include a swap space 206 used by the virtual machine 202. The swap space can be a portion of a hard disk drive, or other persistent storage as used to describe the storage device 112 above. In an example, to free up space in a memory device 104 or the CPU caches 204, the virtual machine 202 or a computing device 100 can transfer data that is not immediately active to the swap space 206 for easy and quicker access compared to other more remote storage areas and devices. In an example, active swap space 206 data can be copied back into the memory device 104 or CPU caches 204. This available swap space 206, while persistent, can be slower than the memory device 104 or the CPU caches 204 can increase the total available system memory of a virtual machine 202 or a computing device 100. Accordingly, data stored in the swap space 206, and used by the virtual machine 202 during a power fail event, can be preserved. In an example, the swap space 206 is already persistent, so no copying of the data is undertaken so long as neither the virtual machine 202 or the computing device 100 copies over or erases the data held there.

[0031] FIG. 3 is a block diagram of an example method for saving and restoring a virtual machine state. This example can be executed in a computing device, for example the computing device 100 of Fig. 1. The method 300 begins at block 302 when a power fail event occurs.

[0032] The power fail event can occur when a long term power source for the computing device 100 becomes removed, damaged, or made non-functioning. A backup or temporary power source can take over powering the computing device until the original power source is fixed, or returns. At block 304, a power fail event is sent in response to the power fail event. The power fail event can be sent to the components of the computing device that have volatile memory as well as to the components of the computing device that have a VMSR 118. The VMSR 118 may perform the method shown and steps involved in saving and restarting the virtual machine.

[0033] At block 306, the processor can be instructed to copy data from the memory, specifically data that was used by the virtual machine, to persistent storage. Similarly, the processor also copies data in the processor to persistent storage. This

data in the processor can include data in any memory bank of the CPU 102 including the L1, L2, and L3 caches.

[0034] At block 308, the processor can be instructed to preserve data in the swap space. As the swap space may already be in a persistent storage device, movement or copying of this data may not be undertaken to preserve it. The processor can indicate an area of the swap space used for virtual machine data at the time of the power fail event, and further prevent overwriting or movement of this data. Any other method of preserving this preserving of swap space data of a virtual machine can also be used, and includes an indication that upon reload, this swap space data will be present in the swap space for the reloading machine.

[0035] At block 310, power is completely lost to a computing device 100. In a complete power loss, the main power supply remains inactive when a backup power element runs out of power and the computing device becomes unpowered. At this point, the data in volatile memory sources can be lost due to lack of power. If the data has been moved to persistent storage, it may be used upon power restore to return the virtual machine state as it was at the time of the power fail event.

[0036] At block 312, a power restore can occur. Although a computing device 100 can be receiving power from a power element such as a backup battery or a secondary power source, a power store in block 312 indicates a return of the primary power source. If a power restore occurs, a power restore event can be sent to components of the computing device 100 including the VMSR 118.

[0037] At block 314, a processor can respond to a power restore event by analyzing data in persistent storage to determine if the virtual machine state can be recovered and restarted. If, yes, the processor determines the virtual machine state can be recovered process flow continues at block 316. If, no, the virtual machine state cannot be recovered, process flow proceeds to block 318.

[0038] At block 316, the recovery of the virtual machine state can proceed. This can include a reloading of data to volatile memory from the persistent storage and reloading a virtual machine state. This can include reloading of data into the CPU 102, CPU caches 204, and other data in the memory device 104. Restoring the virtual machine state can also include ensuring the swap space data from the time of the power fail event is in place as it was at the time of the power fail event.

[0039] At block 318, a processor identifies that the stored data cannot be used to recover the virtual machine state, e.g., a non-recoverable machine state. The identification of the non-recoverability of the machine state can be used in analysis. Any data that was copied from the memory and swap space can be searched for missing components, misdirecting pointers, inconsistent logic, or any other cause of the non-recoverability of the machine state. In an example, this analysis focuses on why the data that recovered could not be used to recover the virtual machine state. This data can be used to improve the recovery process in the future, and can also be used to indicate pieces of the virtual state machine that could be used to perform a partial recovery.

[0040] FIG. 4 is block diagram 400 of an example method for saving a virtual machine during a power fail event. This example method can be implemented in a computing device, such as the computing device 100 of FIG. 1. The example method beings at block 402.

[0041] At block 402, a power element powers a system in response to a power fail event. The system the power element is powering can be the computing device 100 shown in FIG. 1, or any other suitable system. The powering of the system by the power element may have begun prior to the power fail event, can continue to be a power source during a power fail in response to a power fail event. In this way there is no gap in power supply to a system experiencing a power fail. In other examples, the power element can be activated upon a power fail event and the system remains powered during the power up timer period through additional power backup sources such as capacitors, back up batteries, or any other suitable backup power source. The power element of block 402 can be temporary or time limited in the time it will be providing power to a system, before the power element itself runs out of power and the system can become completely unpowered.

[0042] At block 404, a processor can copy data located in the processor and a memory to a persistent storage. The processor in block 404, can be the CPU 102 of FIG. 1. Similarly, the memory can be the memory device 104 of FIG. 1 and the persistent storage can be the storage device 112 of FIG. 1. Other embodiments can be included, where data in a volatile memory can be moved or copied to a non-volatile storage device or powered memory not affected by the power fail. The data copied from memory and from CPUs can include the state of threads and CPUs. In

an example, active threads can be saved as if it was preempted. In this way, any thread within a guest virtual machine can be restarted and restored.

[0043] At block 406, the data located in a swap space of storage is preserved. This preservation can be ensured by a processor such as the CPU 102 of FIG. 1. In another example, the device where the swap space can be located can be immediately unpowered upon the receipt of a power fail event so that data stored in the persistent storage, in including data in the swap space is preserved. A determination can be made by a VMSR 118, in an example, as to whether a persistent storage continuing swap space can be powered down immediately upon receipt of a power fail event or if the persistent storage can be the location of persistent storage of copied data from the processor and memory. These three blocks can indicate the steps taken to save a virtual machine state for recovery upon a power restore to power a system.

[0044] Upon reboot, a new user level application can extract the guest virtual machine state from the data copied from the memory and CPUs and swap space restart the guest virtual machine. This extraction and reassembly of active threads occurs after the system has power restored, the operating system (OS) has booted, and determined that there is a power fail recovery to complete. Postponing the assembly of this data, the threads, and the extraction of the state until after the power restore allows a quicker saving of data that enables this process when the time of data loss due to loss of power is more limited.

[0045] Fig. 5 is a diagram of a an example non-transitory, computer-readable medium that holds code that when executed by a processor saves a virtual machine state on in persistent storage. The computer-readable medium 500 can be accessed by a processor 502 over a system bus 504. In some examples, the code may direct the processor 502 to perform the steps of the current method as described with respect to FIGS. 3 and 4. In some examples, the processor 502 corresponds to a CPU 102 from FIG. 1. Further, the system bus 504 linking the processor 502 with the computer-readable medium 500 can correspond to the bus 106 of FIG. 1 in function and implementation.

[0046] The computer-readable medium 500 can include a power element module 506. The computer-readable medium 500 can include RAM, such as DRAM or SRAM. In some examples, the RAM may be referred to as a logic engine with

program instructions used to store a register table that includes a list of authorized commands for the flash memory device. The power element module can control the function of a power element, which can correspond to the power source 116 and power supply unit 114 of FIG. 1. The power element can be implemented in any way that allows power to be supplied to a system even if a primary power source no longer functions. This can include acting as a backup power supply that activates or takes over upon a power fail event. In some examples, the power element module manages deployment of the power element to ensure constant supply of power until power completely runs out of a system or a primary power source is restored to full functionality.

[0047] The data copier module 508 can direct a processor to copy data located in memory components, particularly data in volatile memory, to a persistent storage. The data copier module 508 can direct a processor 502 to copy data stored in the processor's caches to persistent storage. In an example the persistent storage can be the computer-readable medium 500 upon which the power element module 506 is located. In an example, a portion of data in memory is copied, and the data copier module 508 can choose to copy data in memory that is being used by a virtual machine at the time of a power fail event. In this way the amount of data to be copied can be reduced to match the data for recovering the virtual machine upon power restore and the copy time reduced to increase the odds that the copy completes prior to complete power fail of a system. The power element controlled by the power element module ensures the host computer has a set period of power to allow the other modules to save the virtual machine state.

[0048] In an example, a host computer that supports recovery during a power fail event can also provide the persistent storage for a data copier module 508 can use to save data to be recovered after the power is restored. In an example, this saved data can be used during a host computer power restore to restore the saved state of the virtual machine. The data copied by the data copier module 508 can include a guest virtual machine including a kernel state and host level state that is maintained by host applications. In an example, the infrastructure supporting the guest virtual machine can be saved to persistent storage, in addition to the data in the volatile memory, before power is lost on the host computer.

[0049] In an example, the processor 502 can be a multiprocessor system, in which case CPUs stop executing instructions from the virtual machine and their contexts are saved to non-volatile memory by the data copier module 508. As discussed above, a subset of memory and CPU stored data can be copied, to include the active set of kernel and user physical memory - the memory in use by the virtual machine. This data can also be compressed before storage in non-volatile memory.

[0050] The data preserver module 510 can direct a processor to preserve data located in the swap space of a storage device. In an example, the swap space can be located in a persistent storage corresponding to a storage device 112 from FIG. 1. In an example, the data used by a virtual machine at the time of the power fail event are preserved by the data preserver module 510. In some examples, the swap space may not be used by data for a virtual machine and the swap space can be used in the copying of data from memory or other processes occurring after a power fail event and while a power element acts as a backup power source.

[0051] When the host computer reboots, the data copied by the data copier module 508 and the data preserved by the data preserver module 510 can be parsed together by the processor 502 to extract the saved guest virtual machine image which can then be resumed. In other examples, the automatic saving of data and swap space can be triggered by host computer faults that result in a reboot in addition to the automatic saving of data triggered by power fails. In this example, when the host computer reboots, the host computer can determine whether the fault is recoverable in a process that corresponds to block 314 in FIG. 3.

[0052] The block diagram of FIG. 5 is not intended to indicate that the computer-readable medium 500 is to include the components or modules shown in FIG. 5. Further, any number of additional components may be included within the computer-readable medium 500, depending on the details of the end to end QoS technique and in-band communication described herein.

[0053] While the present techniques may be susceptible to various modifications and alternative forms, the exemplary examples discussed above have been shown only by way of example. It is to be understood that the technique is not intended to be limited to the particular examples disclosed herein. Indeed, the

present techniques include all alternatives, modifications, and equivalents falling within the scope of the appended claims.

CLAIMS

What is claimed is:

1. A system for preserving virtual machine data during a power fail event, the system comprising:

a processor;

a memory;

a storage comprising a swap space used by a virtual machine;

a power element to deliver power to the system during a power fail event; and

wherein, in response to the power fail event, the processor to copy raw data located in the processor and the memory to a persistent storage and to preserve data located in the swap space for use in restoring a transactional state if an application using the raw data was maintaining a transactional state before the power fail event.

2. The system of claim 1, wherein the copied raw data located in the processor and the memory comprises an active set of a virtual machine kernel.

3. The system of claim 1, wherein the raw data copied by the processor to the persistent storage comprises memory in use by the virtual machine, the raw data to be compressed and saved to the persistent storage.

4. The system of claim 1, further comprising the processor to copy raw data located in the processor and the memory to the persistent storage and preserve the raw data located in the swap space in response to a host computer fault.

5. The system of claim 1, further comprising the processor to retrieve and reload a virtual machine state from the persistent storage in response to a power restore event, the virtual machine state comprising raw data copied and preserved by the processor in response to the power fail event.

6. The system of claim 5, further comprising the processor determination of a non-recoverable machine state initiates analysis of the copied raw data to find a cause of non-recoverability.

7. A method for preserving virtual machine raw data during a power fail event, comprising:

powering a system with a power element during a power fail event;

copying, with a processor, raw data located in the processor and a memory to a persistent storage in response to the power fail event for use in restoring a transactional state if an application using the raw data was maintaining a transactional state before the power fail event; and

preserving the raw data located in a swap space of a storage in response to the power fail event for use in restoring a transactional state if an application using the raw data was maintaining a transactional state before the power fail event.

8. The method of claim 7, wherein the raw data copied by the processor to the persistent storage comprises an active set of a virtual machine kernel.

9. The method of claim 7, wherein the raw data copied by the processor to the persistent storage comprises memory in use by a virtual machine, and wherein the raw data to be copied is compressed and saved to the persistent storage.

10. The method of claim 7, comprising a host computer fault initiating the copying and preserving of raw data, the raw data comprising an active set of a virtual machine kernel, raw data on the processor, and also raw data in use by a user in the memory and the swap space.

11. The method of claim 7, comprising retrieving and reloading a virtual machine state from the persistent storage in response to a power restore event, the virtual machine state comprising raw data copied and preserved by the processor in response to the power fail event.

12. The method of claim 11, comprising analyzing the copied raw data and preserved raw data, upon identification of a non-recoverable machine state, for a cause of the non-recoverability.

13. A computer-readable medium having program instructions for preserving virtual machine raw data during a power fail event, the program instructions are executable by a processor to:

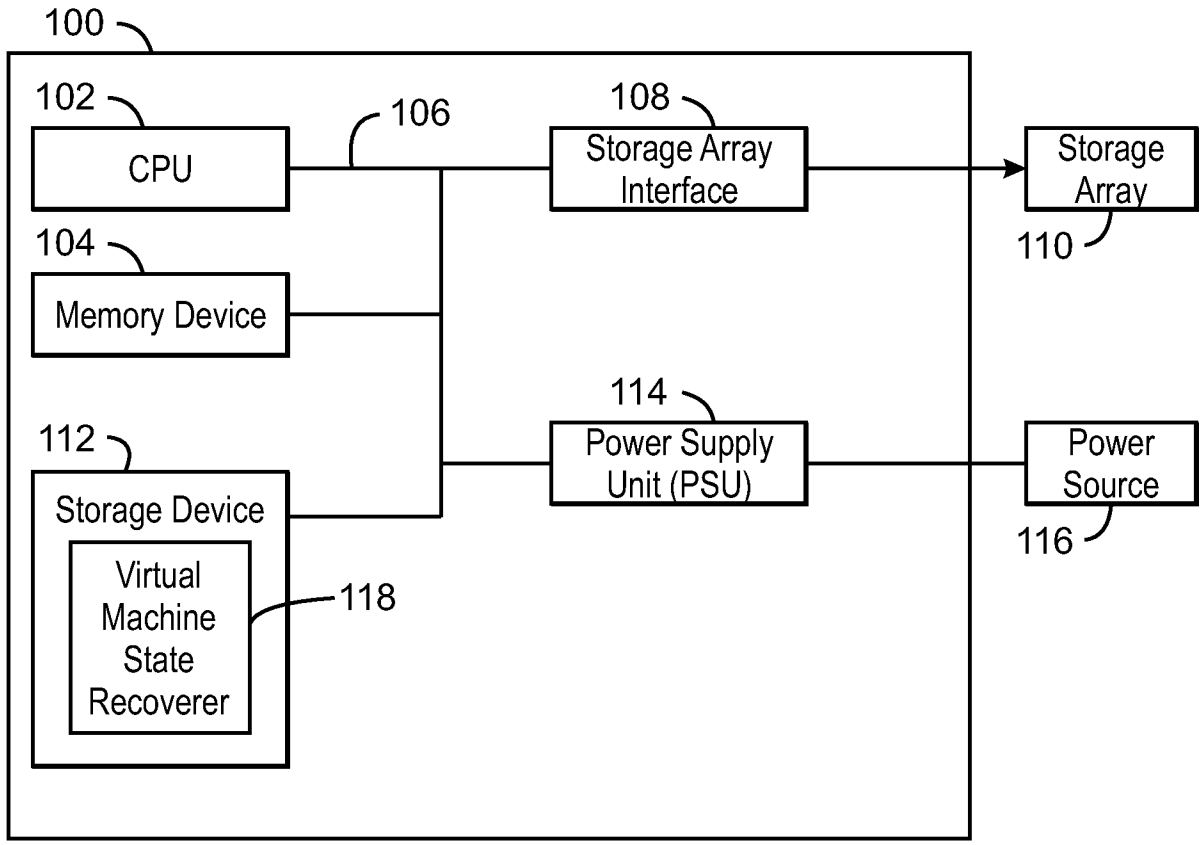
power a system with a power element in response to a power fail event;

copy raw data located in the processor and a memory to a persistent storage in response to the power fail event for use in restoring a transactional state if an application using the raw data was maintaining a transactional state before the power fail event; and

preserve raw data located in a swap space of a storage in response to the power fail event for use in restoring a transactional state if an application using the raw data was maintaining a transactional state before the power fail event.

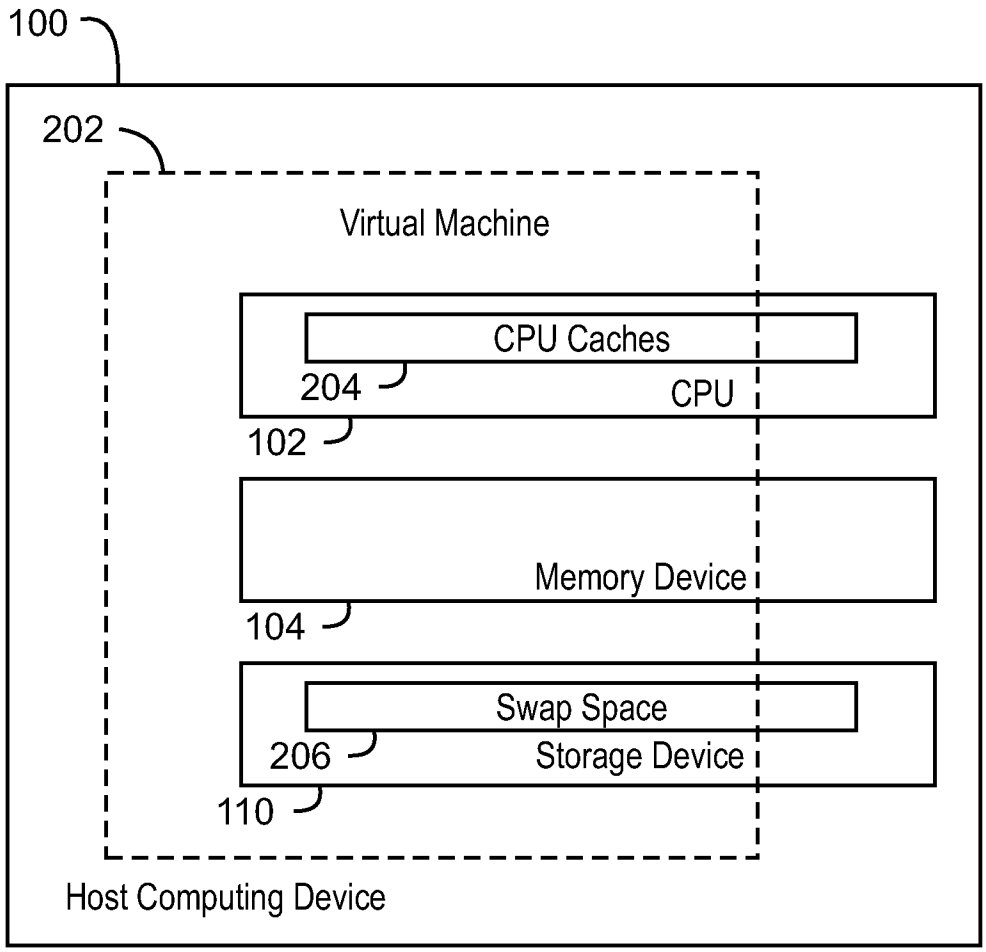
14. The computer-readable medium of claim 13, wherein the raw data copied to the persistent storage comprises an active set of a virtual machine kernel and raw data from the processor and the memory in use by a virtual machine at the time of the power fail event.

15. The computer-readable medium of claim 13, wherein the program instructions, the instructions are executable by a processor to: in response to a power restore event, reload the a virtual machine state from the persistent storage, the virtual machine state comprising raw data copied and preserved by the processor in response to the power fail event.


100
102
CPU
106
108
Storage Array Interface
Storage Array
110
104
Memory Device
114
Power Supply Unit (PSU)
Power Source
116
112
Storage Device
Virtual Machine State Recoverer
118

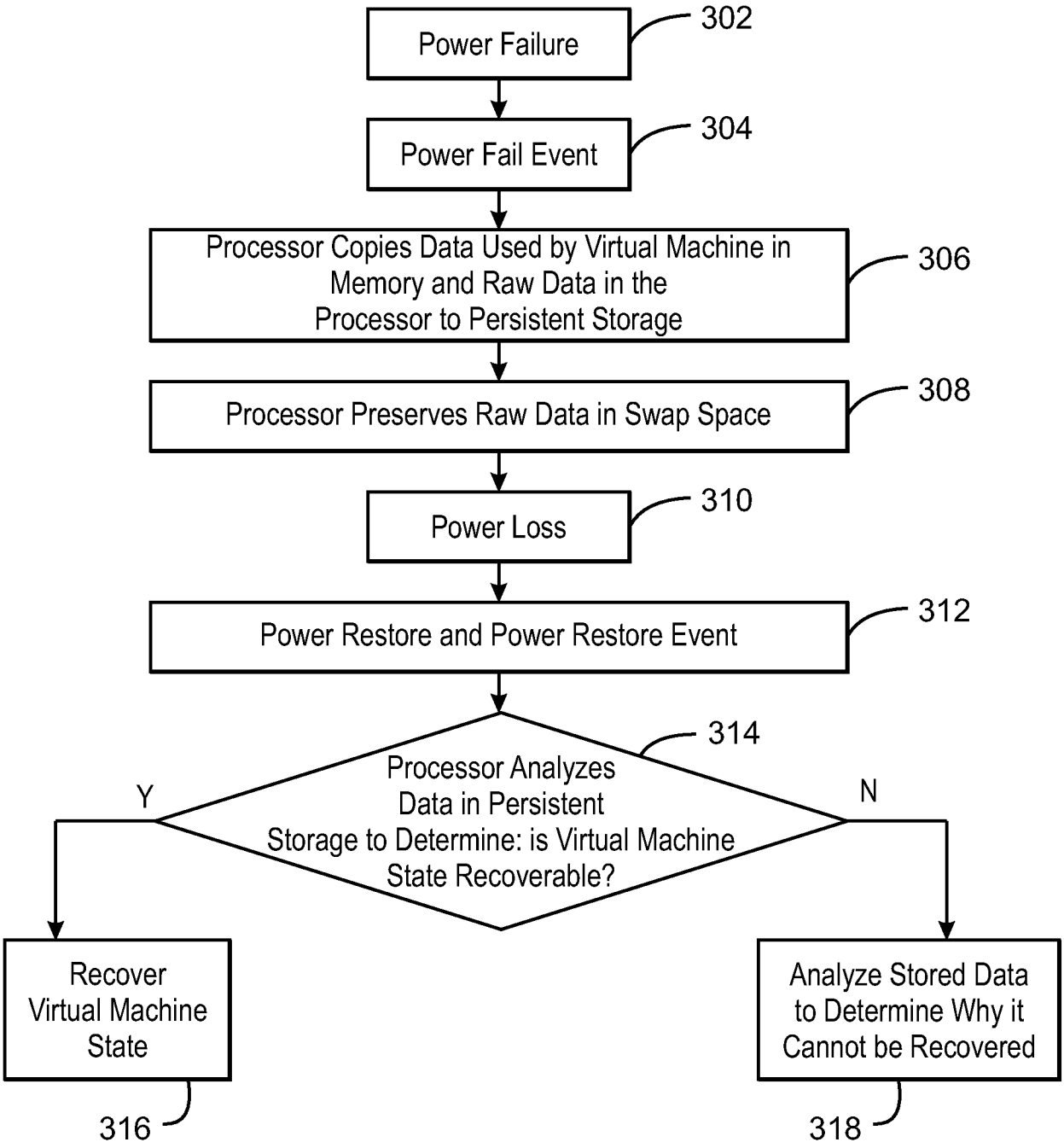
100

FIG. 1


100
202
Virtual Machine
CPU Caches
204
CPU
102
Memory Device
104
Swap Space
206
Storage Device
110
Host Computing Device

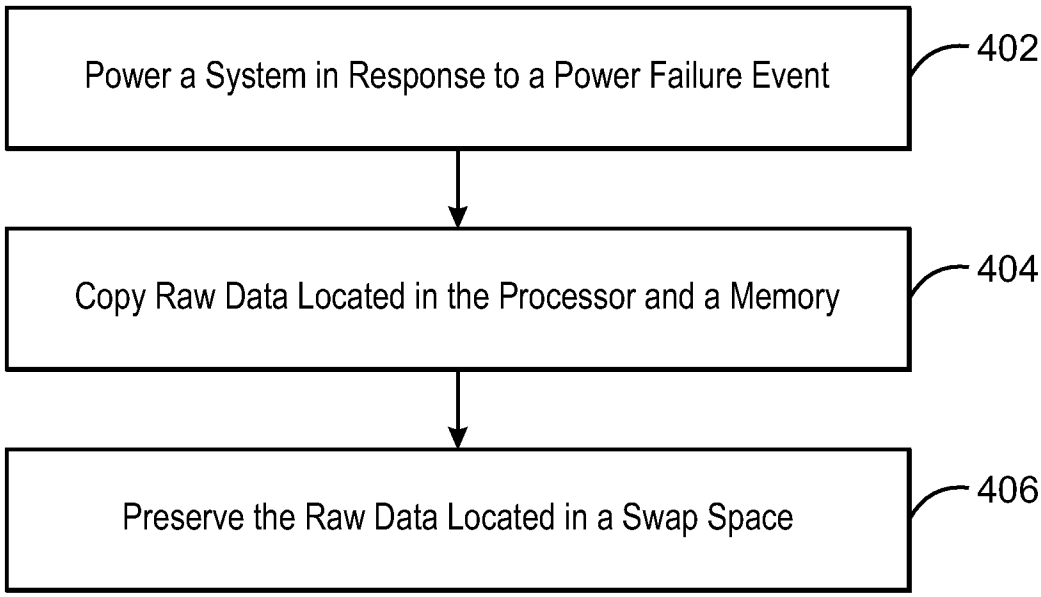
200

FIG. 2


Power Failure
302
Power Fail Event
304
Processor Copies Data Used by Virtual Machine in Memory and Raw Data in the Processor to Persistent Storage
306
Processor Preserves Raw Data in Swap Space
308
Power Loss
310
Power Restore and Power Restore Event
312
Processor Analyzes Data in Persistent Storage to Determine: is Virtual Machine State Recoverable?
314
Y
N
Recover Virtual Machine State
316
Analyze Stored Data to Determine Why it Cannot be Recovered
318

300

FIG. 3


Power a System in Response to a Power Failure Event
402
Copy Raw Data Located in the Processor and a Memory
404
Preserve the Raw Data Located in a Swap Space
406

400

FIG. 4

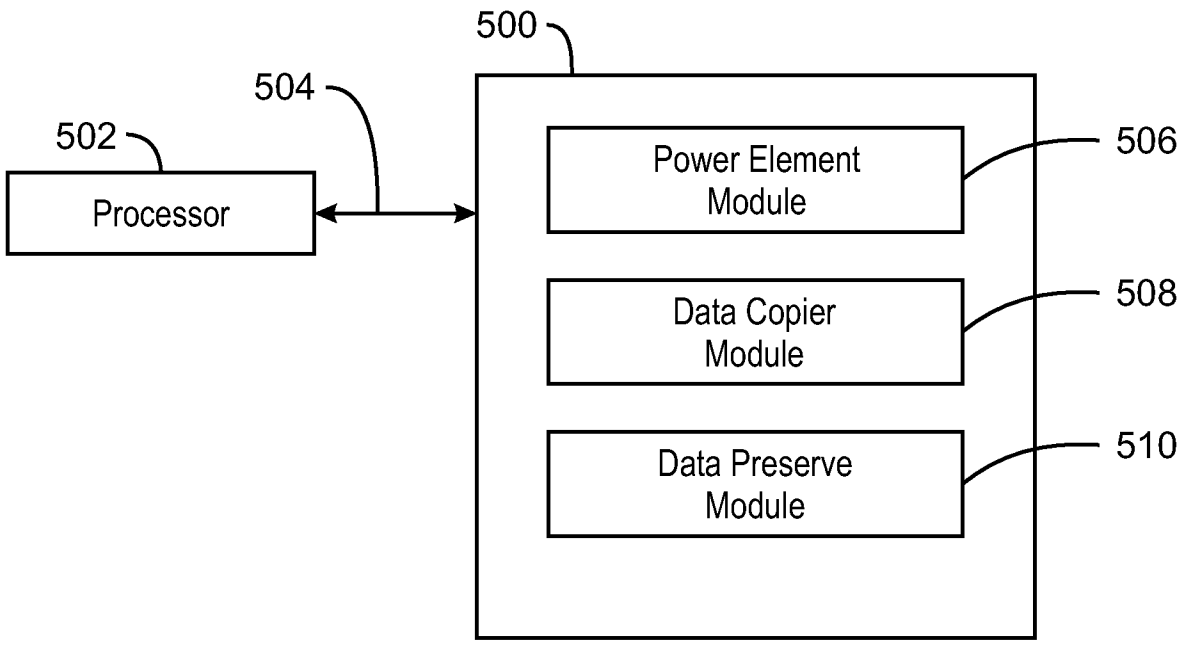

500
504
502
Processor
Power Element Module
506
Data Copier Module
508
Data Preserve Module
510

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/042891

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/455(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 9/455; G06F 12/02; G06F 1/30; G06F 12/00; H04L 9/00; G06F 12/08

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords:preserving, virtual machine, swap space, power element, power fail event, copy, raw data, persistent storage, restoring, transactional state

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2013-0145085 A1 (SUPER TALENT TECHNOLOGY CORP.) 06 June 2013 See paragraph [0073]; claims 6, 7, 12, 13, 17, 21; and figure 2.	1-15
Y	US 2010-0161976 A1 (UTZ BACHER) 24 June 2010 See abstract; paragraph [0068]; claims 1, 6, 7; and figure 1.	1-15
A	US 2015-0058533 A1 (LSI CORPORATION) 26 February 2015 See paragraphs [0037]-[0039]; claims 12, 18; and figures 6, 7.	1-15
A	US 2012-0151118 A1 (DAVID FLYNN et al.) 14 June 2012 See paragraphs [0164]-[0185]; and figure 5A.	1-15
A	US 2006-0136765 A1 (DAVID L. POISNER et al.) 22 June 2006 See paragraphs [0015]-[0022]; and figure 1.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 April 2016 (27.04.2016)	**29 April 2016 (29.04.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/042891

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013-0145085 A1	06/06/2013	CN 103488578 A	01/01/2014
		TW 201426305 A	01/07/2014
		US 8954654 B2	10/02/2015
US 2010-0161976 A1	24/06/2010	US 2014-040418 A1	06/02/2014
		US 8621207 B2	31/12/2013
US 2015-0058533 A1	26/02/2015	US 9021141 B2	28/04/2015
US 2012-0151118 A1	14/06/2012	CN 103262054 A	21/08/2013
		EP 2652623 A2	23/10/2013
		EP 2652623 A4	30/04/2014
		US 2013-346793 A1	26/12/2013
		US 8527693 B2	03/09/2013
		WO 2012-082792 A2	21/06/2012
		WO 2012-082792 A3	11/10/2012
US 2006-0136765 A1	22/06/2006	AT 421119 T	15/01/2009
		CN 101099135 A	02/01/2008
		DE 602005012434 D1	05/03/2009
		EP 1828901 A2	05/09/2007
		EP 1828901 B1	14/01/2009
		JP 2008-522322 A	26/06/2008
		WO 2006-060237 A2	08/06/2006
		WO 2006-060237 A3	14/09/2006