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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
09 May 2019 (09.05.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/084793 A1

(51) International Patent Classification:
G06F 9/50 (2006.01) *H04L 12/46* (2006.01)

(21) International Application Number:
PCT/CN2017/108653

(22) International Filing Date:
31 October 2017 (31.10.2017)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for CN only)*: **NOKIA SHANGHAI BELL CO., LTD.** [CN/CN]; 388 Ning Qiao Road, China (Shanghai) Pilot Free Trade Zone, Shanghai 201206 (CN).

(71) Applicant: **NOKIA SOLUTIONS AND NETWORKS OY** [FI/FI]; Karaportti 3, Espoo, 02610 (FI).

(72) Inventor: **WANG, Cheng**; 388 Ning Qiao Road, China (Shanghai) Pilot Free Trade Zone, Shanghai 201206 (CN).

(74) Agent: **HANHOW INTELLECTUAL PROPERTY**; A Building-1920B, Dinghao Building, No.3 Haidian Street, Haidian, Beijing 100080 (CN).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, 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).

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

(54) Title: A METHOD, APPARATUS AND SYSTEM FOR REAL-TIME VIRTUAL NETWORK FUNCTION ORCHESTRATION

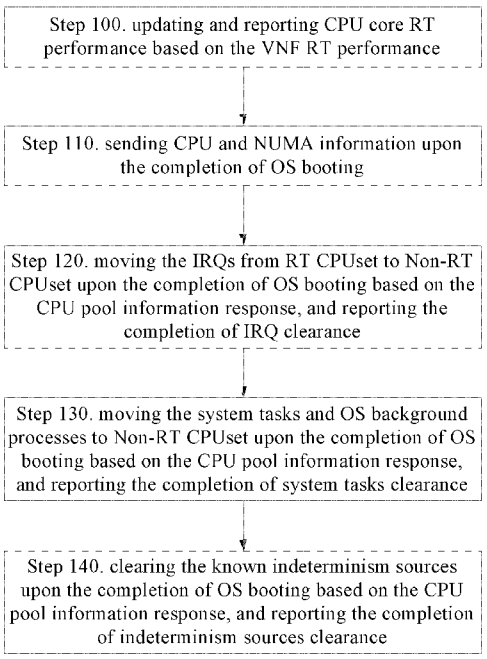

FIG.1

(57) Abstract: Method, apparatus and system for real-time virtual network function orchestration in Real-time Cloud Infrastructure. The method comprises the step of updating and reporting CPU core RT performance; sending CPU and NUMA information; moving the IRQs from RT CPUset to Non-RT CPUset based on the CPU pool information response, and reporting the completion of IRQ clearance; moving the system tasks and OS background processes to Non-RT CPUset based on the CPU pool information response, and reporting the completion of system tasks clearance; clearing the known indeterminism sources based on the CPU pool information response, and reporting the completion of indeterminism sources clearance. The implementation of the method and apparatus improves that with support of NFV, edge cloud can speed new service deployment and achieve resource sharing among different services which allows operators to provision fewer resources.

WO 2019/084793 A1

A METHOD, APPARATUS AND SYSTEM FOR REAL-TIME VIRTUAL NETWORK FUNCTION ORCHESTRATION

FIELD OF THE INVENTION

The present invention relates to a method, apparatus and system for real-time virtual network function orchestration in Real-time Cloud Infrastructure.

BACKGROUND OF THE INVENTION

In recent years, the mobile industry evolves toward IT-ization and cloudization. The underlying idea of Cloud RAN and Network Functions Virtualization (NFV) is to use General Purpose Processor (GPP) to do Radio Access Network (RAN) and CN processing as much as possible. This can exploit the economies of scale of the IT industry and leverage standard IT virtualization technology to consolidate many network equipment types onto industry standard servers (for example, x86 architecture), which could be located in datacenters. With the support of NFV, the mobile network functions are decoupled from hardware which speeds the new service deployment and achieves resource sharing among different services and exploit the advantages provided by cloud infrastructure.

Today`s mobile RAN employs advanced technologies and algorithms to provide high network capacity which requires high processing capability to handle the PHY and MAC layers processing, it is possible to use hardware accelerator to offload some of the PHY compute-intensive functions like Turbo decoder. For example, the current real-time VNF usually contains the following problems:

1. Orchestration of real-time VNF are not covered in traditional Cloud Orchestrator.

One x86 CPU core can afford multiple virtual VNFs. In practical systems, the number of VNFs hosted by a machine can be far larger than the number of CPU cores on a machine. So VNFs have to share a CPU core. Furthermore, the number of VNFs hosted by a machine varies in time due to VNF lifecycle. When a VNF is instantiated, the orchestrator is recommended to determine on which machine the VNF will be placed. Some orchestrators have been developed to deploy cloud applications. Open Stack Heat is a kind of orchestrator which is used for deployment of IT applications like web servers, databases, etc. NFV Tacker is used to deploy VNFs like virtual CPE, CE and PE services. All the above VNFs are non-real-time services. That is, the traditional orchestrators and operating systems only focus on the allocation of computing resources to VNFs, including CPU time and memory. They don't care the real-time (RT) performance of the service very much.

2. Real-time performance of VNF can be impacted by many aspects

In the case where real-time services/VNFs are deployed in cloud environments, a new type of orchestrators that supports deployment of real-time VNFs (RT VNF) is required. Based on practical test, I/O operations can impact real-time performance greatly (The real-time performance of a system can be measured by the interval between the time when an event takes place and the time when the event is served). If the network adapter I/O interrupt thread shares the same CPU core with a real-time application, the real-time performance of the application can be degraded seriously. and the network throughput will also be reduced. The real-time performance for an application sharing CPU core with network adapter interrupt thread is given in Table-1. The results are obtained with network adapter sending packets at rate of 936Mbps. From the Table-1 it is known that the real-time performance cannot meet L2 VNF requirement, as latency larger than 15 us is unacceptable.

Table-1

Timer latency	percentage

>10us	12.83%
>15us	10.79%
>100us	4.28E-4
Max (us)	109.145

For an orchestrator which orchestrates RT VNFs, its orchestrating policy must be different from that for orchestration of traditional VNF which are non-RT applications. The use of CPU core must be carefully planned and the CPU cores for RT VNFs must be isolated from the cores that host I/O interrupt threads. That is, the placement of RT VNFs is recommended to be under the control of new orchestrator which can support RT VNF deployment. After the instantiation of a new RT VNF, the RT performance constraint of both the pre-existing RT VNFs and the newly deployed RT VNFs are recommended to be met.

3. Traditional embedded RT systems and RAN, RT performance monitoring mechanism is not necessary

As these systems run on dedicated hardware appliances which don't involve new VNF instantiation, orchestration and resource sharing/consolidation, once the systems have been adjusted to work well, the RT constraints are always met and RT performance monitoring is unnecessary.

In cloud environment, the number of VNFs hosted by a server varies in time, the processing load of a VNF also varies in time, and the processing capability of CPU cores in the resource pool may be different, a RT performance monitoring mechanism is required.

SUMMARY OF THE INVENTION

In one embodiment, an aspect of this invention relates to a method, apparatus and system for real-time virtual network function orchestration with the method comprising the following steps:

- reporting VNF RT performance
- updating and reporting CPU core RT performance based on the VNF RT

performance;

- sending CPU and NUMA information upon the completion of OS booting;

- moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance;

- moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance;

- clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance.

In another embodiment, an aspect of this invention relates to a method for real-time virtual network function orchestration, with the method comprising the following steps:

- a step of updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information;

- a step of modifying CPUset and sending CPU update command;

- a step of receiving RT VNF deployment request and selecting the target compute node and target CPU.

In another embodiment, an aspect of this invention relates to an apparatus of running on the compute node for real-time virtual network function orchestration, with the apparatus comprising the following modules:

- a module for reporting VNF RT performance;

- a module for updating and reporting CPU core RT performance based on the VNF RT performance;

- a module for sending CPU and NUMA information upon the completion of OS booting;

- a module for moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance;

- a module for moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance;

- a module for clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance.

In another embodiment, an aspect of this invention relates to an apparatus of running on the RT cloud infrastructure for real-time virtual network function orchestration, with the apparatus comprising the following modules:

- a module for updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information;

- a module for modifying CPUset and sending CPU update command;

- a module for receiving RT VNF deployment request and selecting the target compute node and target CPU.

As such, the implementation of this invention improves that with support of NFV, edge cloud can speed new service deployment and achieve resource sharing among different services which allows operators to provision fewer resources. The expected RT cloud infrastructure is able to support RT VNFs deployment and meet the critical RT constraint for RAN processing.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG.1 is a flow chart illustrating a method of the present invention for real-time virtual network function orchestration.

FIG.2 is another flow chart illustrating a method of the present invention

for real-time virtual network function orchestration.

FIG.3 is a block diagram illustrating Real-time Edge Cloud infrastucture architecture.

FIG.4 is a block diagram illustrating Real-time VNF orchestrator software architecture.

FIG.5 is a flow chart illustrating a method of the embodiment for real-time virtual network function orchestration.

FIG.6 is a block diagram illustrating computing resource pool and CPU use planning.

FIG.7 is a flow chart illustrating interactions between VNF orchestrator functions.

FIG.8 is a block diagram illustrating an apparatus of the present invention of running on the compute node for real-time virtual network function orchestration.

FIG.9 is a block diagram illustrating an apparatus of the present invention of running on the RT cloud infrastructure for real-time virtual network function orchestration.

DETAILED DESCRIPTION AND PREFERRED EMBODIMENT

The present invention will now be discussed in detail with regard to the attached drawing figures which are briefly described above. In the following description, numerous specific details are set forth illustrating the applicant`s best mode for practicing the invention and enabling one of ordinary skill in the art of making and using the invention. It will be obvious, however, to one skilled in the art that the present invention may be practiced without many of these specific details. In other instances, well-known machines and method steps have not been described in particular detail in order to avoid unnecessarily obscuring the present invention. Unless otherwise indicated, like

parts and method steps are referred to with like reference numerals.

Referring to FIG.1, an embodiment of a method for real-time virtual network function orchestration comprises:

at step100, updating and reporting CPU core RT performance based on the VNF RT performance;

at step110, sending CPU and NUMA information upon the completion of OS booting;

at step120, moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance;

at step 130, moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance;

at step 140, clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance.

Alternatively, the method for real-time virtual network function orchestration also comprises:

a step of re-arranging IRQs, system tasks and indeterminism source based on the CPUset updating command.

Alternatively, the method for real-time virtual network function orchestration also comprises:

a step of instantiating the VNF based on the VNF instantiation request.

Referring to FIG.2, an embodiment of a method for real-time virtual network function orchestration comprises:

at step200, updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information;

at step210, modifying CPUset and sending CPU update command;

at step220, receiving RT VNF deployment request and selecting the target compute node and target CPU.

Base of the embodiment of a method for realizing dynamic point selection, this invention improves that with support of NFV, edge cloud can speed new service deployment and achieve resource sharing among different services which allows operators to provision fewer resources. The expected RT cloud infrastructure is able to support RT VNFs deployment and meet the critical RT constraint for RAN processing.

In another embodiment, a method of the embodiment for real-time virtual network function orchestration will be described in detail.

This embodiment focuses on two aspects:

1. The construction of real-time cloud infrastructure: This embodiment explicitly distinguishes real-time cloud infrastructure from traditional IT cloud infrastructure. The traditional cloud infrastructure is built for IT applications such as web server and database server, which has poor RT performance. In order to guarantee the RT performance of RT VNFs, the underlying infrastructure is recommended to be able to support RT tasks natively. This requires the compute node in the cloud to be installed with real-time operating system (RTOS) and the cloud resource is recommended to be managed by virtualization platform with good RT performance.

2. With RTOS and RT hypervisor, it is still not enough to support RT VNFs, as too many aspects can impact Linux RT performance. This embodiment proposes a new type of orchestrator for deployment of RT VNFs which is quite different from traditional orchestrator. The proposed orchestrator includes several functions: CPU use planning and CPU isolation, management of source of indeterminism in RT performance, management of system tasks and IRQs, etc. RT VNF orchestration mechanism is also presented which is based on VNF RT performance measurement.

From the preceding discussion, it is known that there is no available orchestrator for real-time VNF orchestration. In order to achieve RAN processing in cloud infrastructure and meet the real-time constraints, real-time cloud infrastructure needs to be constructed and a new type of orchestrator is needed for deployment of RT VNFs in the RT cloud.

With support of NFV, edge cloud can speed new service deployment and achieve resource sharing among different services which allows operators to provision fewer resources. However, implementation of RAN functions in IT server based edge cloud is still very challenging. Although in cloud environment resource sharing between VNFs can improve resource utilization, it can degrade real-time performance as a sequence. How to guarantee RAN function RT performance and let operators take advantages of cloud computing is a tradeoff.

For traditional RAN, the system running on dedicated hardware appliances is a relatively static system with fixed number of RT processes/threads and the maximum processing load is predictable. This system doesn't involve dynamic VNF instantiation, orchestration and resource sharing/consolidation. Once the system has been adjusted to work well, the RT constraints are always met and RT performance monitoring is unnecessary. For Cloud RAN, the VNF may be deployed or destroyed dynamically, as there is limited number of CPU cores in a machine which is far less than the number of VNFs hosted by a machine. Thus, these VNFs have to share CPU core with each other. In this case, it is more challenging to guarantee all the VNFs to meet their RT performances.

As discussed in the Background of The Invention, the mobile industry evolves toward IT-ization and cloudization to use GPP, for example, x86 architecture, to do RAN and CN processing as much as possible. This can exploit the economies of scale of the IT industry and take advantages brought by cloud infrastructure. The general purpose of operating systems and

virtualization platform used by traditional IT cloud cannot support RAN processing due to the poor RT performance.

An RT cloud infrastructure is recommended to be built with a systematic view. the proposed GPP based RT cloud infrastructure includes the following key technologies:

1. Real-time Operating System (RTOS) for compute node in the cloud: we use Linux with PREEMPT_RT patch installed as the RTOS for cloud compute node. PREEMPT_RT is the official patch for Linux kernel which makes Linux gain real-time capabilities. Two methods are available for PREEMPT_RT patch installation, including kernel re-compiling or direct installation of pre-built rpm/deb packages for RT kernel.

2. Real-time virtualization technology/hypervisor: Linux Container-type virtualization technology is adopted in our solution which is a kind of lightweight hypervisor. Linux Container achieves near-native RT performance. In practical system, Docker Container can be used as the virtualization platform as it is more mature than Libvirt Container and OpenVZ.

3. A new type of Orchestrator for RT VNFs orchestration: This new orchestrator includes several functions such as CPU isolation, system tasks management, IRQs management, RT VNF orchestration, etc. CPU isolation technology is very important for RT performance improvement. The CPUs in the cloud is divided into three pools to avoid non-real-time (non-RT) VNFs to compete resources with RT VNFs. One significant difference between the proposed orchestrator and the traditional orchestrator is that the proposed orchestrator monitors the resource usage and VNF performance and deploys VNFs on per-CPU basis, and decides onto which CPU core of which machine the new VNF is recommended to be placed. Traditional orchestrators monitor the resource usage and system performance on per-host basis, it only selects a target machine, but not a target CPU, and it is the local operating system that

plays the main role for task scheduling. It is obvious that the proposed orchestrator schedules the resources and tasks with a finer granularity. This is because careful planning CPU use combing with system tasks and IRQ management can significantly improve VNFs' RT performance. Another important different between the proposed orchestrator and the traditional orchestrator is that the proposed orchestrator's orchestration policy is based on both VNFs' RT performance measurement and CPU utilization, the proposed orchestrator's main task is to guarantee both the newly deployed RT VNF and each of the existing VNFs' RT constraints to be met when these VNFs share a CPU core. It is the proposed orchestrator but not the local OS that determines on which CPU core the VNF will be placed. In the context of traditional orchestrator, there is no RT performance concept, the traditional orchestration policy mainly considers CPU and memory usage which cannot be applied to RT VNF orchestration.

From the hardware point of view, the proposed RT cloud infrastructure uses the same general purpose of hardware platform as the traditional IT cloud, except that some hardware configurations may be different.

All the three key technologies discussed above are indispensable to RT cloud infrastructure which are not required by traditional IT cloud. FIG.3 depicts the proposed RT cloud infrastructure architecture. FIG.4 gives the software architecture for the proposed RT VNF orchestrator which is an important part of the RT cloud infrastructure.

Although the RTOS and Linux Container is introduced as the operating system and virtualization platform, it is still not enough for the compute node to achieve good RT performance. For GPP servers, even the application load is very low, there always exists a large number of daemon processes and system tasks keeping running on the machine and many of them are critical tasks which can degrade system RT performance. Furthermore, management of a large

number of peripheral devices also degrades system RT performance. This is why RT performance of an IT server is more challenging than an embedded system. When constructing GPP based RT cloud infrastructure, it is not simply to stack these technologies together. All aspects that impact VNF RT performance are recommended to be considered by the orchestrator and some policies on how to use cloud infrastructure must be obeyed.

The RT performance of Linux is a very complicated topic, the consequence is that the aspects which impact Linux RT performance are summarized as follows.

- Lower load helps improve RT performance, overloaded can be avoid. According to our observation, if a VNF shares the CPU core with I/O interrupt thread, the RT performance of VNF is bad even though the CPU core has low load. It is the I/O interrupts that introduce unpredictable latency.

- The processor's capability can also impact RT performance, a powerful processor helps improve RT performance, but the RT performance of a system doesn't necessarily to be met even though it has powerful processor. This is why RTOS is needed.

- Task scheduling policy and task priority: Linux provides some scheduling policy for RT applications, such as SCHED_FIFO, SCHED_RR and SCHED_DEADLINE. The RT application can also be set with higher priority than non-RT applications.

- Kernel Preemptibility: With the installation of PREMPT_RT patch, the Linux kernel provides several RT modes, such as Preemptible Kernel (Low-Latency Desktop), Preemptible Kernel (Basic RT), Fully Preemptive Kernel (RT). Fully Preemptible Kernel is preferred for RT VNFs orchestration.

- Task switch overhead: Frequent task switches can introduce overhead and reduce RT performance.

- OS background processes and system tasks: There are a large number of

OS background processes and system tasks running on Linux systems. To avoid resource competition with RT VNFs, these background processes and system tasks can be isolated from the RT VNFs.

- The hardware interrupt request (IRQ), software IRQ and system timers are sources of indeterminism which introduce unpredictable latency. These sources can be managed by the orchestrator.

- CPU load balancer introduced by Linux kernel always tries to evenly distribute tasks over all the available CPUs. This is problematic for RT VNF orchestration, as non-RT tasks could be moved to the CPU cores for RT VNFs, and RT tasks could be moved to CPU core for non-RT tasks. This increases indeterminism in RT performance.

Aiming at the above problems, this invention provides the functions included in the proposed orchestrator which are for tuning the RT performance of the compute nodes and VNF orchestration

FIG.5 illustrates a method for real-time virtual network function orchestration, including the following steps:

At step 500, updating and reporting CPU core RT performance based on the VNF RT performance.

VNF Performance Management Agent collects the local resource usage of each compute node, especially focuses on the RT performance reports sent by VNFs. The updated RT performance and CPU utilization are then forwarded to VNF Manager.

At step 510, sending CPU and NUMA information upon the completion of OS booting.

When adding a new compute node in the cloud, the local CPU Pool Management Agent reports the CPU core number and NUMA node number to the CPU Pool Manager.

At step 520, updating CPU pool based on the CPU and NUMA

information, and synchronizing the CPU pool information.

As the OS background processes, system tasks and non-RT VNFs can compete CPU time with the RT VNFs and introduce uncertain impact on RT performance, these processes can be isolated from each other. In this embodiment, it is categorized into three kinds of processes for RAN processing: RT processes, non-RT processes and DPDK processes. The GPP computing resources are also divided into three kinds of groups/pools corresponding to the three kinds of processes.

- CPU pool for RT VNFs: MAC scheduler is such an RT VNF which includes downlink and uplink scheduler entities. The RT performances of RT VNFs running in this kind of CPU Pool are monitored with the certain mechanism.

- CPU pool for Non-RT VNFs: Not all RAN VNFs require high RT performance. For example, timer events of RRC are usually on the order of tens of milliseconds. Compared to PHY and MAC processing, this kind of processes can be considered as non-RT VNFs. In practical system, the background processes and system tasks also can be placed in this pool. The reason why there has a dedicated CPU pool for non-RT processes is that RT processes usually have higher priority than non-RT processes. In the case where RT VNFs share a CPU with non-RT VNFs, the non-RT VNFs may be throttled if the load of RT VNFs is high.

- CPU pool for DPDK processes: DPDK is widely used in products which is a set of drivers and libraries for fast packet processing. DPDK requires dedicated CPU cores for packet receiving and processing in order to achieve high throughput. So, there has a dedicated CPU pool for DPDK processes/threads.

The CPU Pool Manager is responsible for the maintenance of all the CPU pools, increase or decrease CPU cores in a pool as the machine may be

powered on/off on demand and the available CPU cores varies in time. The CPU Pool Manager has global view of the use of CPU core. Each kind of pools spans across multiple compute nodes as shown in FIG.6. There is an entry for each CPU core of a machine in the pool. When the VNF manager needs to deploy a VNF, it asks the CPU Pool Manager for the available machine list for RT VNFs or non-RT VNFs, according the type of VNF to be deployed.

When receiving the CPU core number and NUMA node number, CPU Pool Manager divides CPU cores of the new compute node into different pools. CPU Pool Manager provides core information for other network elements to facilitate RT performance tuning on a compute node.

At step 530, moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance.

Although IRQ load balancing is disabled by Indeterminism Source Management Agent, this cannot prevent some IRQs from being initially placed on the CPUs in RT CPUset during system booting, which have serious negative impact on RT performance. These IRQs can be handled by the CPUset for non-RT or background processes. IRQ Management Agent is responsible for moving these IRQs from RT CPUset to non-RT CPUset. The affinity of these IRQs can be controlled using the /proc file system. Assuming CPU0 and CPU1 are for non-RT tasks, the default affinity is first set to CPU0 or CPU1 to make sure that new interrupts won't be handled by the RT CPUs. The set {CPU1, CPU0} is represented as a bitmask set to 3, (0000,0011B)

echo 3 > /proc/irq/default_smp_affinity

Then move IRQs to the non-RT CPUset

echo 3 > /proc/irq/<irq>/smp_affinity

All active IRQs can be found in file /proc/interrupts. When moving the

IRQs, the IRQ Management Agent is recommended first to query the CPU Pool Manager to obtain the CPU index for non-RT tasks on this compute node.

At step 540, moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance.

For a compute node to host RT tasks, even though with the management of indeterminism source, sometimes system tasks still can be observed on the CPUs in the RT CPUset after system is booted. System Tasks Management Agent is responsible for moving system tasks from the CPUs in the RT CPUset to the CPUs in non-RT CPUset or to the CPUs in the background processes CPUset if it exists. Run the following command to move system tasks.

echo pid_of_task > /sys/fs/cgroup/cpuset/nonrt/tasks

The process IDs of these system tasks can be found in pseudo-file /sys/fs/cgroup/cpuset/rt/tasks.

At step 550, clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance.

For a Linux system, there are many sources which introduce indeterminism in RT performance. Indeterminism Source Management Agent is responsible for configuring or modifying Linux system parameters so that its RT properties become more deterministic. The main management work of Indeterminism Source Management Agent is briefly described below:

- Disable CPU load balancer and define CPUset: In the default setting, Linux's task scheduler is free to migrate tasks to evenly distribute the processing load among the available CPUs. That might be good for throughput, but it could damage RT performance. To turn off automatic load balancing and statically assign tasks to CPUs can increase determinism. At least three kinds

of method can be used to assign a task to a specific CPU, including sched_setaffinity() system call, taskset command and CPUset mechanism. For example, when the VNF Manager has selected a proper CPU core on a compute node to place the RT VNF, the ID of the target CPU core is transferred to the VNF as an argument, then the VNF can use sched_setaffinity() to pin the RT threads to the given CPU core. The Linux kernel CPUset mechanism can also be used to control the processor placement and memory placement of processes. In practice, at least 3 CPUsets can be defined. One is for RT VNFs, one is for DPDK processes, the other is for non-RT VNFs. By default, load balancing is done across all CPUs, except those marked isolated using the kernel boot time "isolcpus" option. The CPU cores can be isolated for RT VNFs and DPDK from the CPU cores for non-RT VNFs, background processes and system tasks by setting the "isolcpus" option in file grub.cfg. The use of CPUs in CPU pools for RT VNFs and DPDK is better to be under the control of the proposed RT VNF Manager, but not the OS scheduler. This can be achieved by setting the pseudo-file cpuset.sched_load_balance to 0 which disables the automatic load balancing over the allowed CPUs in the defined CPUsets (the load balancing in root CPUset also can be disabled). Furthermore, to avoid non-RT tasks to use RT CPUset, it is needed to make the CPUs in the RT CPUset exclusive by setting RT cpuset's pseudo-file cpuset.cpu_exclusive set to 1. The local CPU core lists marked by "isolcpus" option is recommended to be synchronized with the CPU Pool Manager so that the CPU Pool Manager has the global view of CPU cores use.

- Management of NUMA memory node: In NUMA system, the CPU accesses to its own memory node is faster than other memory node. The RT CPUset and non-RT CPUset need to be associated with their own memory nodes. In the case where there are more than two NUMA memory nodes, if the

CPU cores assigned to RT VNFs belong to different NUMA node, it is better to create multiple CPUsets for the RT tasks and each RT CPUset is associated to its own memory node. This helps improve RT performance of RT VNFs. The locally defined CPUset information is recommended to be synchronized with the CPU Pool Manager. The following command associates NUMA node 2 with RT CPUset and make NUMA node two exclusive to the RT CPUset.

echo 2 > /sys/fs/cgroup/cpuset/rt/cpuset.mems

echo 1 > /sys/fs/cgroup/cpuset/rt/cpuset.mem_exclusive

- Management of IRQ affinity: By default, Linux enables the interrupt request (IRQ) load balancing service which evenly distributes IRQs across all the CPUs in the system. If an IRQ is serviced on the CPU which is currently executing real-time VNFs, the CPU has to switch contexts which when combined with cache misses can cause tens of microseconds of latency. By stopping this service (IRQ balance) it allows us to control on which CPU interrupts will run. In practical system, we can configure the mask in the smp_affinity file and assign certain IRQs, for example, interrupts from SCSI controller or Ethernet card, to be handled by specific CPUs. The CPUs can be selected from the non-RT CPU pool.

- Disable CPU frequency scaling: By default, Linux enable dynamic CPU frequency scaling in order to reduce power consumption. But this technique can affect the system's RT properties. For CentOS, two methods can be used to disable CPU frequency scaling. 1. edit /etc/default/grub to include the line GRUB_CMDLINE_LINUX_DEFAULT= "intel_pstate=disable" and run grub2-mkconfig -o /boot/grub/grub.cfg. 2. recompile Linux kernel without CPU frequency scaling option.

- Management of RT throttling mechanism: To prevent the RT applications scheduled as SCHED_FIFO or SCHED_RR from consuming all CPU power, an RT throttling mechanism is used by Linux kernel to limit the

amount of CPU power that the RT tasks can consume. The default setting for this mechanism is that RT tasks can consume up to 95% of CPU power of a machine. This can be changed by writing the new number to files:

/proc/sys/kernel/sched_rt_runtime_us

and

/proc/sys/kernel/sched_rt_period_us

- Disable memory overcommit: By default, the Linux kernel allows applications to allocate more memory than is actually available in the system. The idea of memory overcommit is to provide a more efficient memory usage, under the assumption that processes typically ask for more memory than they will actually need. Overcommitting means there is a risk if processes try to utilize more memory than is available. If this happens, the kernel invokes the Out-Of-Memory Killer to scan through the task list and selects a task to kill to reclaim memory. In this case, the whole system may become unresponsive for a significant amount of time which is unacceptable for RT VNFs. Memory overcommit can be disabled by the following command:

echo 2 > /proc/sys/vm/overcommit_memory

- Offload RCU callbacks: The Read-Copy-Update (RCU) system is a lockless mechanism for mutual exclusion inside the kernel which improves data sharing among threads. As a consequence of performing RCU operations, callbacks, done as a soft IRQ by default, are queued on CPUs to be performed at a future moment when removing memory is safe. This adds unpredictable latencies to application. RCU callbacks can be offloaded using the “rcu_nocbs” and “rcu_nocb_poll” kernel boot parameters. To remove one or more CPUs from the candidates for running RCU callbacks, specify the list of CPUs in the “rcu_nocbs” kernel parameter, for example: “rcu_nocbs=4-6” means that RCU callbacks will not be done on CPU4, CPU5 and CPU6.

- Set TSC boot parameter: The time stamp counter is a per-CPU counter

for producing time stamps. Since the counters might drift a bit, Linux will periodically check that they are synchronized. By telling Linux with boot parameter "TSC=reliable" that the counters are reliable, Linux will no longer perform the periodic synchronization. This improves Linux RT performance.

- Remove vmstat timer: vmstat timer is used for collecting virtual memory statistics.The statistics are updated at an interval specified as seconds in file /proc/sys/vm/stat_interval. The amount of jitter can be reduced by writing a large value to this file. However, that will not solve the issue with worst-case latency. Linux kernel version 3.12 or newer removes the periodic statistics collection and replaces it with a solution that only triggers if there is actual activity that needs to be monitored.

- BDI writeback affinity: Since block I/O can have a serious negative impact on RT performance, it is recommended to be moved to the non-RT CPUset. Two steps are needed:

Disable NUMA affinity for the writeback threads

echo 0 > /sys/bus/workqueue/devices/writeback/numa

Assuming CPU0 and CPU1 are in the non-RT CPUset, set the affinity to the CPUset

echo 3 > /sys/bus/workqueue/devices/writeback/cpumask

- Disable machine check: The x86 architecture has a periodic check for corrected machine check errors. The periodic machine check requires a timer that causes jitter. The periodic check can be disabled on the RT CPUs. For each CPU in the RT CPUset, do the following:

echo 0 >/sys/devices/system/machinecheck/machinecheck<cpu>/check_interval

echo 0 >/sys/devices/system/machinecheck/machinecheck2/check_interval

echo 0 >/sys/devices/system/machinecheck/machinecheck3/check_interval

- Disable the watchdog: The watchdog timer is used to detect and recover from software faults. It requires a regular timer interrupt which is a jitter

source. This interrupt can be removed at the cost of less error detection. The watchdog can be disabled at compile time or in runtime as follows:

echo 0 > /proc/sys/kernel/watchdog

- Increase flush time to disk: To make writebacks of dirty memory pages occur less often than the default, we can do the following:

echo 1500 > /proc/sys/vm/dirty_writeback_centisecs

- Network queues affinity: If applications need to send or receive network traffic, some timers are created for network protocols on the specific CPUs. If there is a need of network traffic only on the non-RT applications, network queues affinity can be set as follows to improve RT properties:

echo <NRT cpus mask> > /sys/class/net/<ethernet interface>/queues/<queue>/<x/r>ps_cpus

All the above management and configuration modifications can be done by Indeterminism Source Management Agent in an automatic way. To ensure the kernel boot parameter modifications take effects, the compute node needs to reboot.

At step 560, receiving RT VNF deployment request and selecting the target compute node and target CPU.

In a practical system, for a given type of VNF, the maximum processing load is usually known which can be used to estimate if the new VNF can be accommodated on a CPU core. The following steps can be used by VNF Manager to select the target CPU core.

- Select a set of CPU cores with better RT performance from the CPU pool for RT VNFs;

- Obtain the types and the numbers of VNFs hosted by each CPU core in the set;

- Calculate the potential maximum processing load on each CPU core in the set according the types and the numbers of VNFs;

- Recalculate the potential maximum processing load on each CPU core in the set assuming the new VNF is deployed on it;

- Select a subset of CPU cores from the set which can accommodate the new VNF's maximum processing load and has higher margin;

- Select the CPU core with the best RT performance from the subset as the target CPU.

Once the target CPU core is selected, the ID of the target CPU core will be transferred to the target compute node. the new VNF will be instantiated on the target CPU core under the control of local VNF Performance Management Agent.

The difference between two timestamp counters T_{rep} and T_{exp} is used to measure the RT performance. For the VNF with multiple threads, only one thread needs to measure the interrupt latency. The timestamp T_{local} is different from T_{rep} and T_{exp}, it is gotten from the local compute node's system clock, not from the PCIe device. For a VNF with multiple threads, T_{local} is recommended be gotten by the thread which is responsible for getting T_{rep}. The difference between two contiguous T_{local} (T_{local2} - T_{local1}) can also be used to monitor the RT performance. If the difference approaches one millisecond, it means the processing load on this CPU core is very high which is recommended not to accept new VNF anymore.

FIG.7 illustrates the interactions between these functions of the proposed orchestrator. RT VNFs periodically report their RT performances to local VNF Performance Management Agent. Performance Management Agent summaries these reports and extracts the RT performances (for example, the worst RT performance of a VNF hosted by a CPU core) for each CPU core and reports to RT VNF Manager. After completion of compute node booting, the local CPU Pool Management Agent reports the local CPU and NUMA information to CPU Pool Manager, CPU Pool Manager makes decision on which CPUs to

be added to RT CPUset, non-RT CPUset and DPDK CPUset respectively, then the IRQ Management Agent sends request to CPU Pool Manager to obtain the local CPUset information and then moves the IRQs from the RT CPUset to non-RT CPUset. System tasks Management Agent sends requests to CPU Pool Manager and moves system tasks and background processes to the non-RT CPUset according to the received CPUset information. Based on the CPUset information, Indeterminism Source Management Agent changes the IRQ affinity, RCU callback affinity, BDI writeback affinity, etc. Once the local CPUset is changed by CPU Pool Manager under certain condition, all these agents need to re-arrange the affinities according to the new CPUset. When a new RT VNF needs to be deployed, RT VNF Manager selects the target compute node and the target CPU and sends the orchestration parameters to the local Performance Management Agent. Under the control of Performance Management Agent, the VNF is instantiated on the target CPU core.

This embodiment describes how to select target machine and target CPU. This is just an example for orchestration policy. It is proposed that the orchestrators monitor VNF RT performance, and the orchestration policy may be different depending on how to use the measured RT performance.

FIG.8 illustrates an embodiment of an apparatus of running on the compute node for real-time virtual network function orchestration, including the following modules:

a module for updating and reporting CPU core RT performance based on the VNF RT performance (updating and reporting module 800);

a module for sending CPU and NUMA information upon the completion of OS booting (sending module 810);

a module for moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response,

and reporting the completion of IRQ clearance (IRQ moving module 820);

a module for moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance (system tasks moving module 830);

a module for clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance (clearing module 840).

Alternatively, the said apparatus comprises:

a module for re-arranging IRQs based on the CPUset updating command;

a module for re-arranging system tasks based on the CPUset updating command; and

a module for re-arranging indeterminism source based on the CPUset updating command.

Alternatively, the said apparatus comprises:

a module for instantiating the VNF based on the VNF instantiation request.

FIG.9 illustrates an embodiment of an apparatus of running on the RT cloud infrastructure for real-time virtual network function orchestration, including the following modules:

a module for updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information (updating module 900);

a module for modifying CPUset and sending CPU update command (modifying module 910);

a module for receiving RT VNF deployment request and selecting the target compute node and target CPU (receiving and selecting module 920).

Alternatively, the said apparatus comprises:

a module for sending CPU pool information response based on the CPU pool information request.

Alternatively, the said apparatus comprises:

a module for sending CPU updating command.

Alternatively, the said apparatus comprises:

a module for sending VNF instantiation request.

The specific implementation functions of each module contained in the above-mentioned apparatus of running on the compute node for real-time virtual network function orchestration and apparatus of running on the RT cloud infrastructure for real-time virtual network function orchestration have been described in the previous method embodiment and are not repeated here.

Note that in the embodiment of this invention, each module is divided according to function of logic, but is not limited to the above, as long as can realize the corresponding function; In addition, the specific name of each functional module is only for the purpose of making it easy to distinguish, and it is not used to limit the arrange of protection of this invention.

Note that the invention is not limited to the embodiments described hereinabove, but extends to all the embodiments that are in accordance with its idea. The alternatives or options described in this part stem directly from the description of the preceding technological steps. They are valid for the illustrative applications such as micro-batteries but can be transposed to other microelectronic components. Unless mentioned otherwise, the steps that describe the examples presented in each part are based on the same principles mentioned beforehand.

CLAIMS

1. A method for real-time virtual network function orchestration, comprising these steps of:

a step of updating and reporting CPU core RT performance based on the VNF RT performance;

a step of sending CPU and NUMA information upon the completion of OS booting;

a step of moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance;

a step of moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance;

a step of clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance.

2. The method of claim 1, wherein the said method comprises:

a step of re-arranging IRQs, system tasks and indeterminism source based on the CPUset updating command.

3. The method of claim 1, wherein the said method comprises:

a step of instantiating the VNF based on the VNF instantiation request.

4. A method for real-time virtual network function orchestration, comprising these steps of:

a step of updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information;

a step of modifying CPUset and sending CPU update command;

a step of receiving RT VNF deployment request and selecting the target

compute node and target CPU.

5. The method of claim 4, wherein the said method comprises:

a step of sending CPU pool information response based on the CPU pool information request.

6. The method of claim 4, wherein the said method comprises:

a step of sending CPU updating command.

7. The method of claim 4, wherein the said method comprises:

a step of sending VNF instantiation request.

8. An apparatus of running on the compute node for real-time virtual network function orchestration, comprising:

a module for updating and reporting CPU core RT performance based on the VNF RT performance;

a module for sending CPU and NUMA information upon the completion of OS booting;

a module for moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance;

a module for moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance;

a module for clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance.

9. The apparatus of claim 8, wherein the said apparatus comprises:

a module for re-arranging IRQs based on the CPUset updating command;

a module for re-arranging system tasks based on the CPUset updating command; and

a module for re-arranging indeterminism source based on the CPUset

updating command.

10. The apparatus of claim 8, wherein the said apparatus comprises:

a module for instantiating the VNF based on the VNF instantiation request.

11. An apparatus of running on the RT cloud infrastructure for real-time virtual network function orchestration, comprising:

a module for updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information;

a module for modifying CPUset and sending CPU update command;

a module for receiving RT VNF deployment request and selecting the target compute node and target CPU.

12. The apparatus of claim 11, wherein the said apparatus comprises:

a module for sending CPU pool information response based on the CPU pool information request.

13. The apparatus of claim 11, wherein the said apparatus comprises:

a module for sending CPU updating command.

14. The apparatus of claim 11, wherein the said apparatus comprises:

a module for sending VNF instantiation request.

15. A system for real-time virtual network function orchestration, comprising: the said apparatus of running on the compute node for real-time virtual network function orchestration from claim 8 to 10 and the said apparatus of running on the RT cloud infrastructure for real-time virtual network function orchestration from claim 11 to 14.

16. A computer readable storage medium, storing the computer code, when the computer code is executed, the method of claim 1 to 3 or claim 4 to 7 is executed.

17. A computer program product, when the computer program product is executed, the method of claim 1 to 3 or claim 4 to 7 is executed.

18. A computer product, comprising:

one or more processors;

storage of storing one or more computer programs;

when the one or more computer programs are executed by the one or more processors, the one or more processors implement the method of claim 1 to 3 or claim 4 to 7.

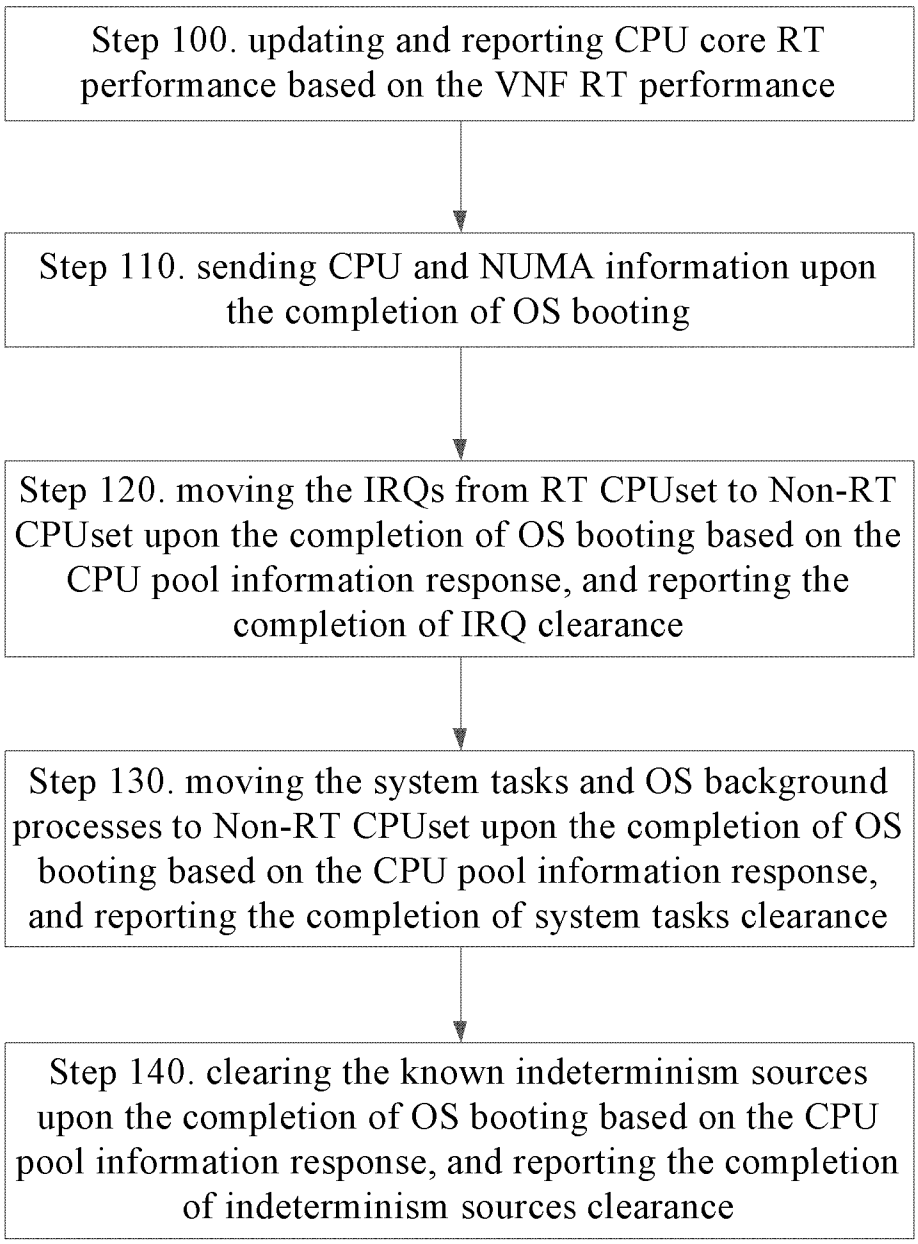
Step 100. updating and reporting CPU core RT performance based on the VNF RT performance
Step 110. sending CPU and NUMA information upon the completion of OS booting
Step 120. moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance
Step 130. moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance
Step 140. clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance

FIG.1

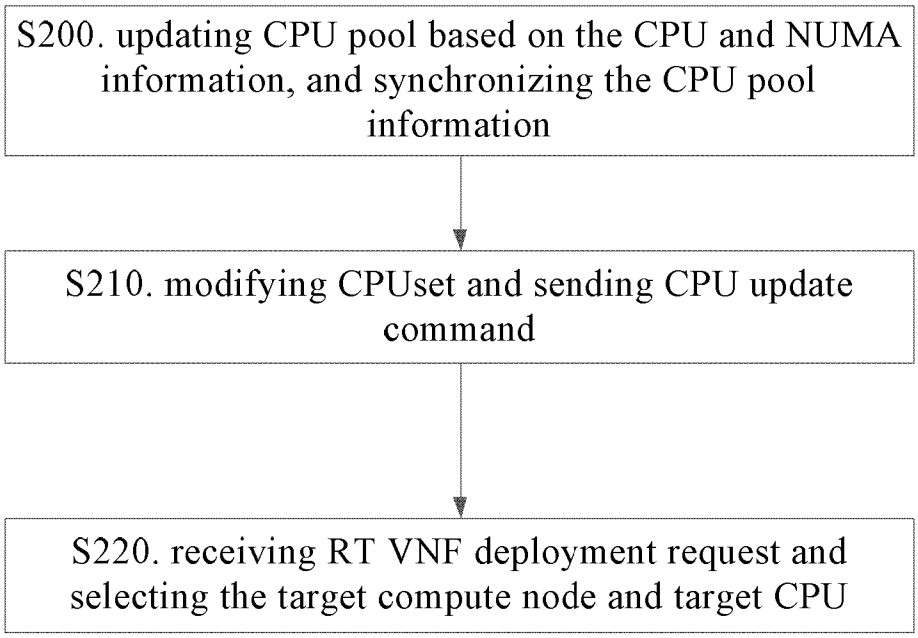
S200. updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information
S210. modifying CPUset and sending CPU update command
S220. receiving RT VNF deployment request and selecting the target compute node and target CPU

FIG.2

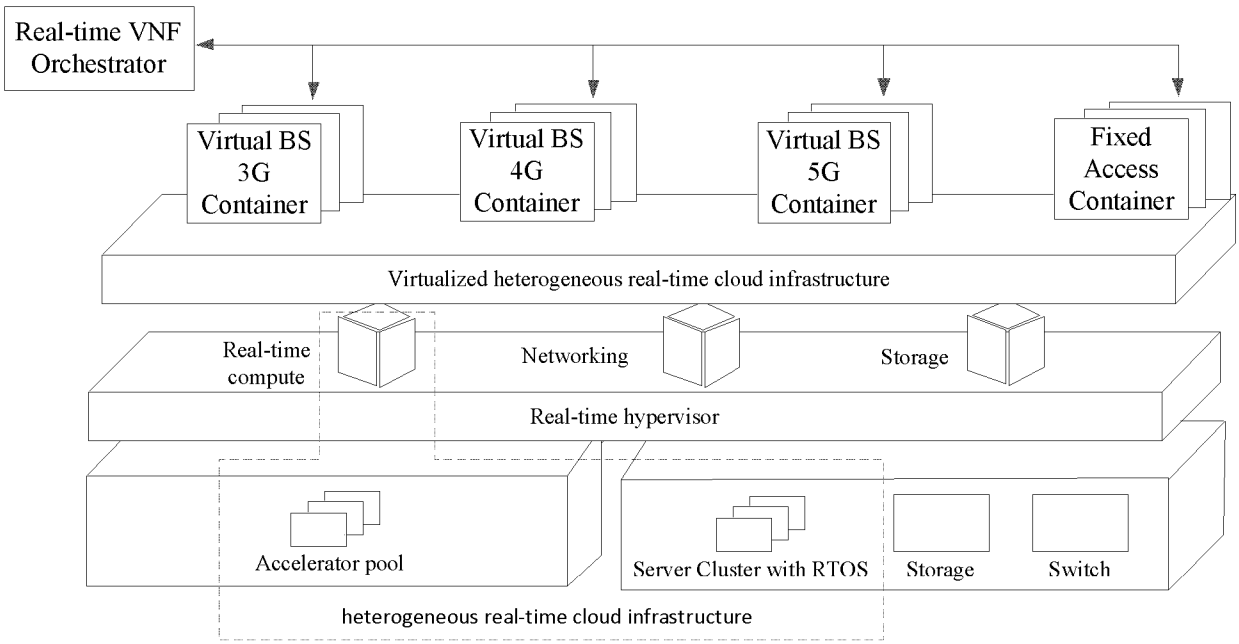
Real-time VNF Orchestrator
Virtual BS 3G Container
Virtual BS 4G Container
Virtual BS 5G Container
Fixed Access Container
Virtualized heterogeneous real-time cloud infrastructure
Real-time compute
Networking
Storage
Real-time hypervisor
Accelerator pool
Server Cluster with RTOS
Storage
Switch
heterogeneous real-time cloud infrastructure

FIG.3

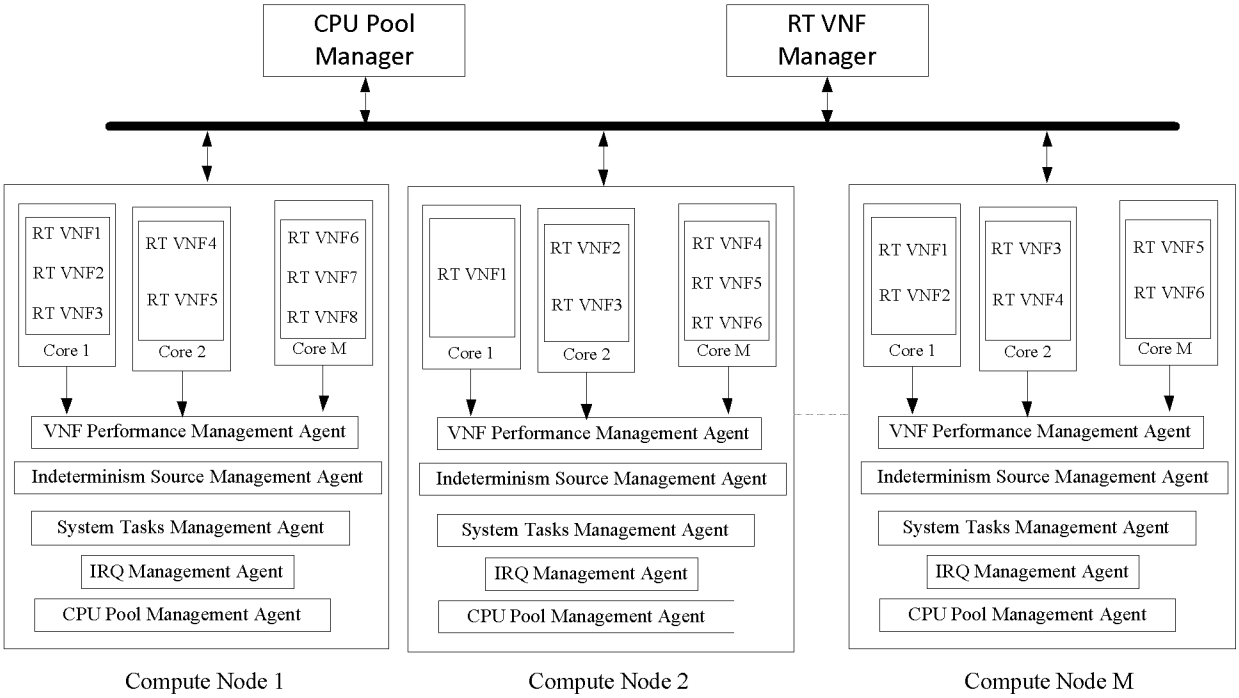
CPU Pool Manager
RT VNF Manager
RT VNF1
RT VNF2
RT VNF3
Core 1
RT VNF4
RT VNF5
Core 2
RT VNF6
RT VNF7
RT VNF8
Core M
VNF Performance Management Agent
Indeterminism Source Management Agent
System Tasks Management Agent
IRQ Management Agent
CPU Pool Management Agent
Compute Node 1
RT VNF1
Core 1
RT VNF2
RT VNF3
Core 2
RT VNF4
RT VNF5
RT VNF6
Core M
VNF Performance Management Agent
Indeterminism Source Management Agent
System Tasks Management Agent
IRQ Management Agent
CPU Pool Management Agent
Compute Node 2
RT VNF1
RT VNF2
Core 1
RT VNF3
RT VNF4
Core 2
RT VNF5
RT VNF6
Core M
VNF Performance Management Agent
Indeterminism Source Management Agent
System Tasks Management Agent
IRQ Management Agent
CPU Pool Management Agent
Compute Node M

FIG.4

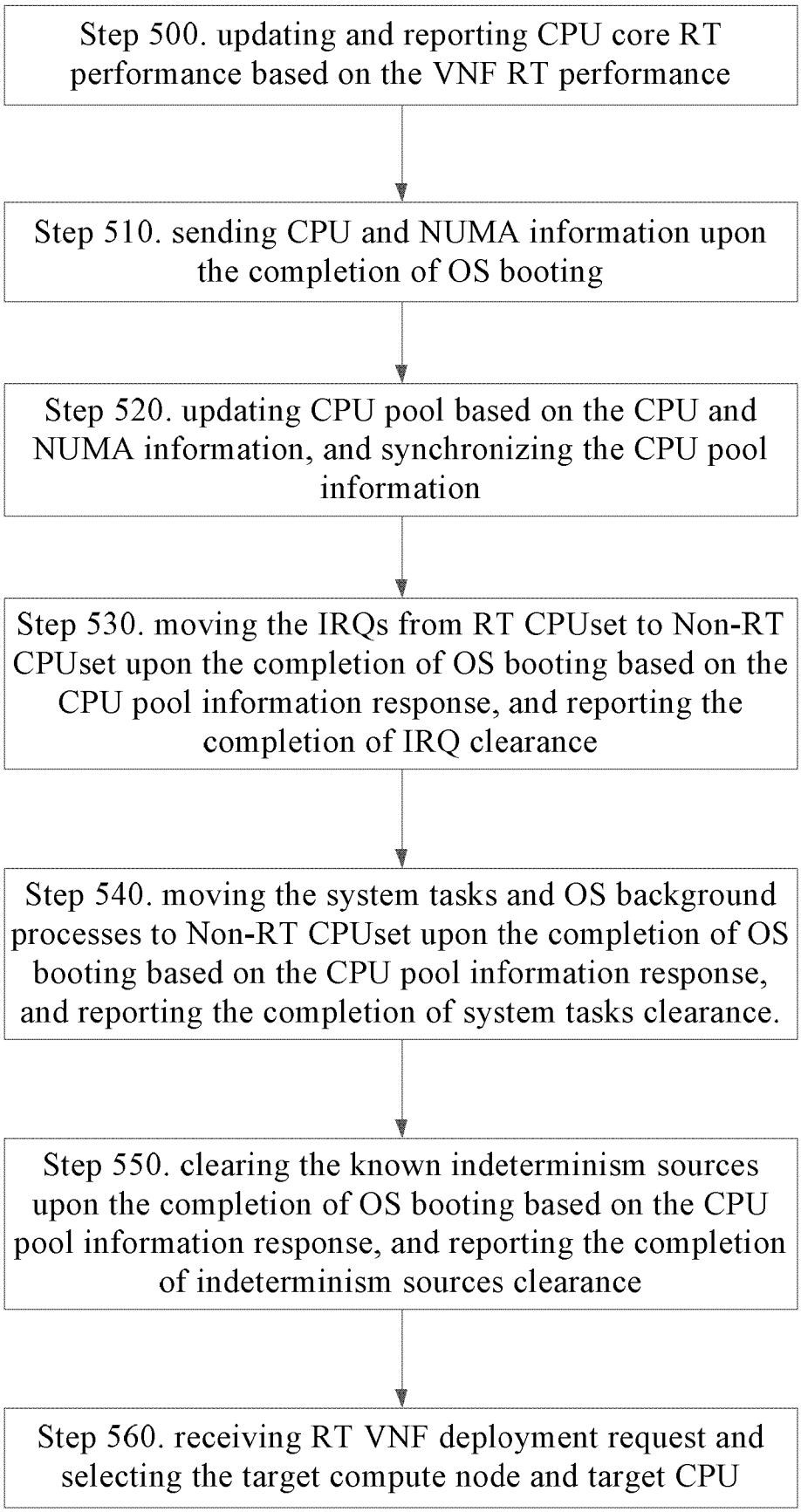
Step 500. updating and reporting CPU core RT performance based on the VNF RT performance
Step 510. sending CPU and NUMA information upon the completion of OS booting
Step 520. updating CPU pool based on the CPU and NUMA information, and synchronizing the CPU pool information
Step 530. moving the IRQs from RT CPUset to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of IRQ clearance
Step 540. moving the system tasks and OS background processes to Non-RT CPUset upon the completion of OS booting based on the CPU pool information response, and reporting the completion of system tasks clearance.
Step 550. clearing the known indeterminism sources upon the completion of OS booting based on the CPU pool information response, and reporting the completion of indeterminism sources clearance
Step 560. receiving RT VNF deployment request and selecting the target compute node and target CPU

FIG.5

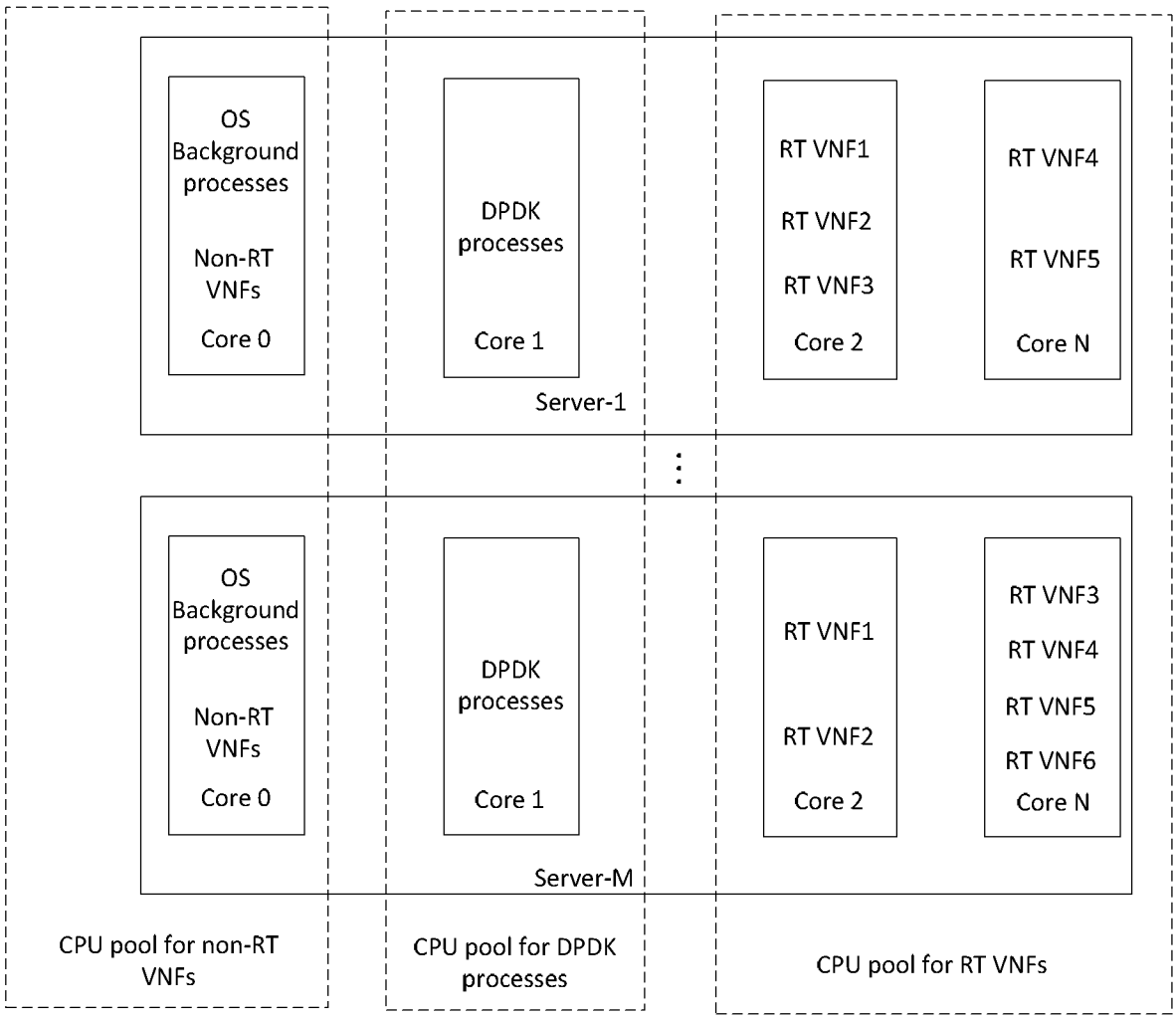
OS Background processes
Non-RT VNFs
Core 0
DPDK processes
Core 1
RT VNF1
RT VNF2
RT VNF3
Core 2
RT VNF4
RT VNF5
Core N
Server-1
OS Background processes
Non-RT VNFs
Core 0
DPDK processes
Core 1
RT VNF1
RT VNF2
Core 2
RT VNF3
RT VNF4
RT VNF5
RT VNF6
Core N
Server-M
CPU pool for non-RT VNFs
CPU pool for DPDK processes
CPU pool for RT VNFs

FIG.6

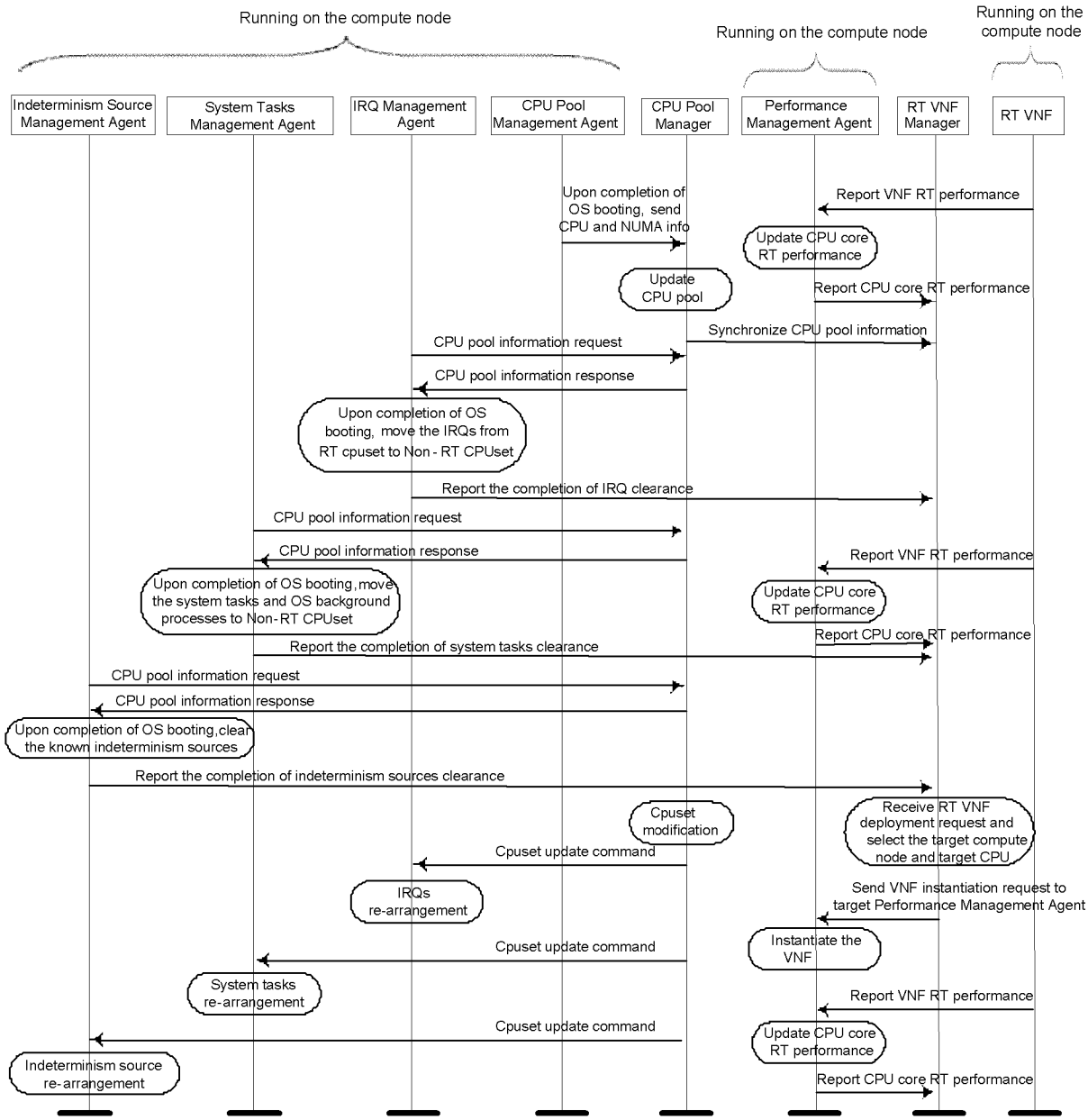
Running on the compute node
Running on the compute node
Running on the compute node
Indeterminism Source Management Agent
System Tasks Management Agent
IRQ Management Agent
CPU Pool Management Agent
CPU Pool Manager
Performance Management Agent
RT VNF Manager
RT VNF
Upon completion of OS booting, send CPU and NUMA info
Report VNF RT performance
Update CPU core RT performance
Update CPU pool
Report CPU core RT performance
Synchronize CPU pool information
CPU pool information request
CPU pool information response
Upon completion of OS booting, move the IRQs from RT cpuset to Non - RT CPUset
Report the completion of IRQ clearance
CPU pool information request
CPU pool information response
Report VNF RT performance
Upon completion of OS booting, move the system tasks and OS background processes to Non-RT CPUset
Update CPU core RT performance
Report CPU core RT performance
Report the completion of system tasks clearance
CPU pool information request
CPU pool information response
Upon completion of OS booting, clear the known indeterminism sources
Report the completion of indeterminism sources clearance
Cpuset modification
Receive RT VNF deployment request and select the target compute node and target CPU
Cpuset update command
IRQs re-arrangement
Send VNF instantiation request to target Performance Management Agent
Instantiate the VNF
Cpuset update command
System tasks re-arrangement
Report VNF RT performance
Cpuset update command
Update CPU core RT performance
Indeterminism source re-arrangement
Report CPU core RT performance

FIG.7

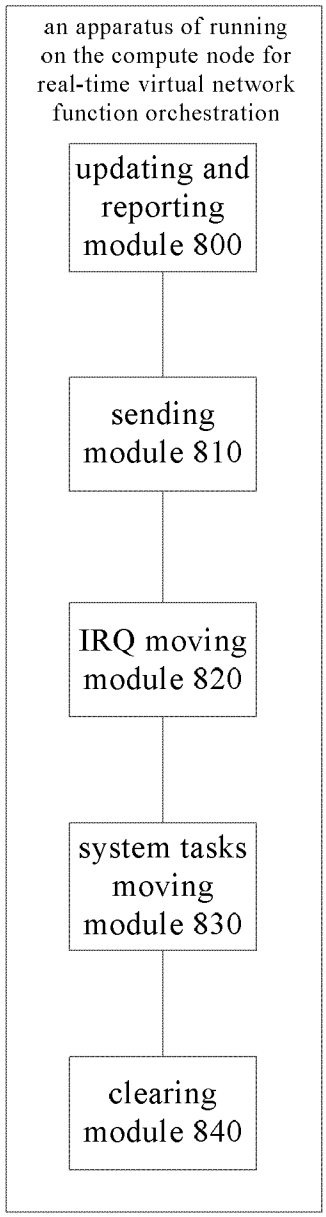
an apparatus of running on the compute node for real-time virtual network function orchestration
updating and reporting module 800
sending module 810
IRQ moving module 820
system tasks moving module 830
clearing module 840

FIG.8

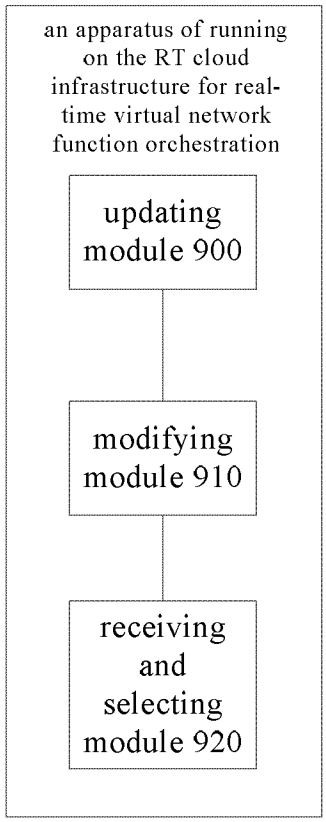
an apparatus of running on the RT cloud infrastructure for real-time virtual network function orchestration
updating module 900
modifying module 910
receiving and selecting module 920

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2017/108653

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/50(2006.01)i; H04L 12/46(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)

H04L; G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT;WPI;EPODOC;CNKI;3GPP;GOOGLE:virtual network function, network function virtualization, VNF, NFV, CPU, compute unit, computer process, centre process, central process, real time, RT, request, require, select, determine, choose, chosen, confirm, decide, affirm, validate, pool, set, IRQ, system task, background process, orchestrator, orchestration, memory, storage, store

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 105634782 A (HUAWEI TECHNOLOGIES CO., LTD.) 01 June 2016 (2016-06-01) the whole document	1-18
A	CN 106533723 A (ZTE CORP.) 22 March 2017 (2017-03-22) the whole document	1-18
A	WO 2017058274 A1 (INTEL IP CORP.) 06 April 2017 (2017-04-06) the whole document	1-18
A	ETSI. "Network Function Virtualisation (NFV); Management and Orchestration" *ETSI GS NFV-MAN 001 v1.1.1*, 31 December 2014 (2014-12-31), pages 1-184	1-18

☐ 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
08 July 2018	**06 August 2018**

Name and mailing address of the ISA/CN	Authorized officer
STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China**	**NING,Bo**
Facsimile No. **(86-10)62019451**	Telephone No. **86-(10)-53961584**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2017/108653

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 105634782 A	01 June 2016	WO 2016070729 A1	12 May 2016
CN 106533723 A	22 March 2017	WO 2017041556 A1	16 March 2017
WO 2017058274 A1	06 April 2017	None	