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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 March 2017 (30.03.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/052935 A1

(51) International Patent Classification:
H04L 12/24 (2006.01) *H04L 12/931* (2013.01)

(21) International Application Number:
PCT/US2016/048478

(22) International Filing Date:
24 August 2016 (24.08.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/866,567 25 September 2015 (25.09.2015) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 14/866,567 (CON)
Filed on 25 September 2015 (25.09.2015)

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).

(72) Inventors: **HERDRICH, Andrew J.**; 2111 NE 25th Avenue, Hillsboro, Oregon 97124 (US). **CONNOR, Patrick L.**; 7853 SW 174th PL, Beaverton, Oregon 97007 (US). **KUMAR, Dinesh**; 16878 NW Desert Canyon Drive, Beaverton, Oregon 97006 (US). **MIN, Alexander W.**; 5566 NW Primino Ave, Portland, Oregon 97229 (US). **DAHLE, Daniel J.**; 7256 SW Lynnwood Court, Wilsonville, Oregon 97070 (US). **SOOD, Kapil**; 1647 NW 191st Place, Beaverton, Oregon 97006 (US). **SHAW, Jeffrey B.**; 109 Oak Mill Road, Folsom, California 95630 (US). **VERPLANKE, Edwin**; 5000 W Chandler Blvd, Chandler, Arizona 85226 (US). **DUBAL, Scott P.**; 772 NW Autumncreek Way, Apt. O-201, Beaverton, Oregon 97006 (US). **HEARN, James R.**; 1221 NE 51st Ave #310, Hillsboro, Oregon 97124 (US).

(74) Agents: **PERDOK, Monique M.** et al.; Schwegman Lundberg & Woessner, P.A., c/o CPA Global, 900 Second Avenue South, Suite 600, Minneapolis, Minnesota 55402 (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,

[Continued on next page]

(54) Title: OUT-OF-BAND PLATFORM TUNING AND CONFIGURATION

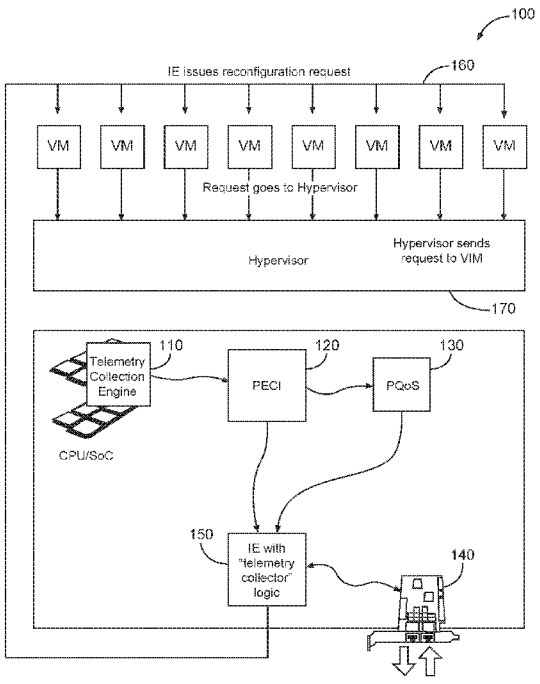

Fig. 1

(57) Abstract: Devices and techniques for out-of-band platform tuning and configuration are described herein. A device can include a telemetry interface to a telemetry collection system and a network interface to network adapter hardware. The device can receive platform telemetry metrics from the telemetry collection system, and network adapter silicon hardware statistics over the network interface, to gather collected statistics. The device can apply a heuristic algorithm using the collected statistics to determine processing core workloads generated by operation of a plurality of software systems communicatively coupled to the device. The device can provide a reconfiguration message to instruct at least one software system to switch operations to a different processing core, responsive to detecting an overload state on at least one processing core, based on the processing core workloads. Other embodiments are also described.

WO 2017/052935 A1

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))*

OUT-OF-BAND PLATFORM TUNING AND CONFIGURATION

CLAIM OF PRIORITY

[0001] This patent application claims the benefit of priority to U.S. Application Serial No. 14/866,567, filed September 25, 2015, which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] Embodiments described generally herein relate to management of resources in operator networks. Some embodiments relate to dynamic allocation of networking resources and tuning and monitoring of resource allocation.

BACKGROUND

[0003] Current cloud datacenters have been experiencing a large scale-up and scale-out for I/O devices, and this is causing new challenges for datacenter manageability, reliability and performance. Telemetry can assist datacenter software with workload placement and mapping, but providing this telemetry can place a further drain on datacenter resources.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] In the drawings, which are not necessarily drawn to scale, like numerals may describe similar components in different views. Like numerals having different letter suffixes may represent different instances of similar components. The drawings illustrate generally, by way of example, but not by way of limitation, various embodiments discussed in the present document.

[0005] FIG. 1 illustrates components of a platform for providing out-of-band telemetry in accordance with some embodiments.

[0006] FIG. 2 illustrates a device including telemetry collection logic for providing out-of-band telemetry in accordance with some embodiments.

[0007] FIG. 3 is a flow diagram of an initializing and benchmarking phase of a network interface card (NIC) affinization algorithm in accordance with some

embodiments.

[0008] FIG. 4 is a flow diagram of an operational phase of a NIC affinization algorithm in accordance with some embodiments.

[0009] FIG. 5 illustrates a network function virtualization (NFV) system architecture and data flows in accordance with some embodiments.

[0010] FIG. 6 is a block diagram of a system for out-of-band platform configuration parameter configurability in accordance with some embodiments.

[0011] FIG. 7 illustrates performance versus cache occupancy of a cache-sensitive workload.

[0012] FIG. 8 illustrates performance versus cache sensitivity of a compute-bound workload that does not exhibit sensitivity to cache resources.

[0013] FIG. 9 is a flow diagram of an example hardware-implemented method for implementing a performance monitoring and aggregation algorithm in accordance with some embodiments.

[0014] FIG. 10 illustrates cache sensitivity data for a cache-sensitive workload that can be analyzed for making configuration decisions in accordance with some embodiments.

DETAILED DESCRIPTION

[0015] Recently, datacenter operators have experienced challenges in providing large-scale manageability, reliability and performance for I/O devices, such as Ethernet 10/40/100 Gbps/++ devices, Infiniband devices, RSA Optical fabric/interconnects, switches, etc. Additionally, as operators scale up/out, guaranteeing the performance of individual network flows or types of traffic becomes ever more difficult, particularly in network cloud implementations such as European Telecommunications Standards Institute (ETSI) Network Functions Virtualisation (NFV) and software defined network (SDN) Cloud. Still further, network operators and service providers demand high levels of resiliency with network cloud systems. To complicate the situation even more, the range in features, capabilities and performance between deployed server systems increases and heterogeneity increases as customers add newer machines to their deployed fleets without necessarily retiring older machines.

[0016] I/O scale out/up can be achieved and better managed overall in cloud

datacenters through mechanisms that rely on reliable and continuous delivery of telemetry from the platform I/O devices (e.g. network interface cards (NICs), switches, etc.) to external automatic orchestration logic, for a more flexible and software-defined infrastructure. However, providing such telemetry can place a further drag on operator systems, such that it becomes ever more difficult to comply with service level agreements (SLAs).

[0017] Embodiments provide an orchestration controller that processes continuous streams of telemetry, both actively and passively, to manage network-centric workloads by assigning workloads to specific platforms, and migrating between platforms, as overload conditions or other adverse conditions are detected. By maintaining context and assisting with workload placement and mapping to specific platforms in accordance with various embodiments, operators can spend fewer resources, both in terms of time and instrumentation, directly managing workload placement on complex heterogeneous fleets of servers. Embodiments therefore provide solutions for issues associated with large-scale scale up/out management of reliability and performance. Embodiments can also provide benefits to compute-centric platforms.

[0018] Additionally, methods and systems in accordance with various embodiments provide for improved synchronization and accurate telemetry across servers within racks, across the datacenter, and across multiple datacenters spanning multiple geographical locations. Such synchronization is an issue in datacenter operations where it is important that the user always observes the latest copies of data.

[0019] Synchronization issues apply to not just time but configuration state as well. Configuration state can include many different parameters including power management aggressiveness, system feature state (e.g., reliability, availability and serviceability (RAS) feature dynamic configuration), and shared resource monitoring/allocation configuration for Intel® Resource Director Technology (RDT), or Platform QoS or PQoS). Any or all of these technologies can enable monitoring and control over shared platform resources such as last-level cache space, memory bandwidth, and in the future I/O bandwidth.

[0020] Some datacenter operators maintain time and configuration state synchronization using an in-band approach, where involvement from the

operating system (OS) or virtual machine manager (VMM) running on the system is provided to accept and apply updated parameters such as a time value or a configuration setting. This involvement from the OS or VMM introduces overhead and latency by interrupting the normal operation of the OS/VMM and consuming compute cycles. By offloading these tasks to out-of-band (OOB) systems in accordance with various embodiments, collection, aggregation and analysis of data can be performed without the use of Intel® Architecture (IA) cores or other cores implementing an OS or VMM.

[0021] Although some embodiments use the Intel® Management Engine (ME) or Innovation Engine (IE), other instantiations are possible in various other embodiments that use other combinations of OOB-capable microcontrollers and firmware that is capable of receiving parameters from an external source and applying them to update the current system configuration.

[0022] While it is possible to perform OOB management and synchronization tasks using the ME and supporting software, open sample code for the IE can also be provided to datacenter operators to accomplish the OOB platform tuning and optimization, allowing tuning parameters and even tuning algorithms to be modified by datacenter operators in accordance with their needs.

Platform Telemetry Driven Network Function Deployments for Operator Networks

[0023] As briefly mentioned earlier herein, I/O scale out/up can be achieved in cloud datacenters through mechanisms that rely on reliable and continuous delivery of telemetry from the platform I/O devices (e.g. network interface cards (NICs), switches, etc.) to an external automatic orchestration logic. However, providing such telemetry can place a further drag on operator systems, such that it becomes ever more difficult to comply with service level agreements (SLAs).

[0024] Embodiments address these and other concerns by providing delivery of SLA services, fault management, alarms, and high availability on Cloud systems. The telemetry in accordance with various embodiments follows a tenant enforced secure reception and delivery of telemetry using, by way of nonlimiting example, Intel ® Software Guard Extensions (SGX), Trusted Platform Module (TPM), or a secure Trusted Execution Environment (TEE).

Ongoing industry efforts at Open Platform for NFV (OPNFV) and ETSI NFV are directed to defining formal requirements for these usages.

[0025] Embodiments provide the capability for Intel ® IE, the OOB Core, Intel ® ME, or other deployments, platforms and software to reconfigure or access physical or virtual NICs. Embodiments provide OOB or side channel access to the NICs without disrupting in-band accesses from the Intel Architecture® (IA) cores running the NIC drivers.

[0026] In contrast to some telemetry-provisioning systems, a telemetry agent in accordance with various embodiments collects data from NICs, in addition to other data described herein. Embodiments provide service quality metrics in accordance with provisioning of SLA requirements, as specified by the ETSI NFV standards group, which specifies the need for detailed NIC, I/O and platform telemetry. Messaging, OOB telemetry, metrics and periodicity described in accordance with various embodiments may be used for meeting Operator ETSI NFV requirements on IA-based NFV platforms such as Open-Source OPNFV and Sunrise Trail platforms. Some categories of this telemetry can include virtual machine (VM) operations or virtual network function (VNF) operations (e.g., latency, VM Clock error, VM Dead on Arrival, etc.), virtual network operations (e.g., packet delays, delay variations, network outages, port status, policy integrity, etc.). The telemetry agent of various embodiments processes data from processor cores, chipset, memory, the platform, NICs, storage, virtual switches (vSwitches), acceleration units (e.g., encryption, compression, etc.).

[0027] Devices in accordance with various embodiments can calculate or generate SLA metrics. For example, some devices will check that SLA is within acceptable limits, or enforce SLA violations by reporting violations to an orchestrator or other operator. Devices can also provide audit capabilities, which in the context of OOB removes the need to notify application software of adverse conditions, SLA violations, changes, etc.

[0028] The OOB approach of various embodiments can enhance or improve performance debugging, because the OOB approach does not add introspection overhead to a system already running near peak capacity. Accordingly, embodiments can avoid skewing of the performance results. OOB or side

channel access to the NICs, in accordance with various embodiments, avoids disrupting in-band accesses from the IA cores running the NIC driver. Accordingly, embodiments can reduce overheads and interrupt rates for reconfiguration.

[0029] Available Ethernet I/O exposes only a limited set of telemetry, and embodiments specify additional telemetry exposed by I/O adapters (including virtual I/O adapters like vSwitch), which can be accessible by out of band techniques in accordance with various embodiments.

[0030] In some network-workload centric operator deployments, the I/O device is directly assigned to the network-centric workloads (often referred to as Virtual Network Function – VNF), and has little or no intervention from the hypervisor/VMM. In such deployments, CPU cores or threads are assigned to these VNFs and cannot be used for telemetry and orchestration, because latency requirements of the VMs are sufficiently stringent that VM traffic cannot be paused. Accordingly, the OOB mechanism of various embodiments may be desirable because such OOB mechanisms can run concurrently and asynchronously with the VMs while not occupying or contending for platform resources that the VMs are depending on to meet their latency/bandwidth targets. In addition, the mechanisms may be able to enforce SLA by, for example, administering ports (aggregation, disaggregation) and allowing/restricting traffic to match established SLAs.

[0031] In another embodiment, multiple VNFs may be running on the platform and each may be assigned to one or more physical or virtual I/O devices. The OOB mechanism thus becomes a comprehensive mechanism for telemetry and orchestration across all workloads and across all devices on the platform.

[0032] FIG. 1 illustrates components of a platform 100 for providing OOB telemetry in accordance with some embodiments. The platform 100 includes a Telemetry Collection Engine 110 that collects information from CPU/SoC. The Telemetry Collection Engine 110 can be located in the platform controller (PCH) in the south complex which runs separately which from a socket perspective from other applications, although embodiments are not limited to implementation in the PCH.

[0033] A Platform Environment Control Interface (PECI) 120 passes on CPU

telemetry to device 150 for reconciliation. The device 150 can include or be included in Intel ® IE although embodiments are not limited thereto. PECI is a protocol for collecting the data described herein, although embodiments are not limited to use of PECI.

[0034] PQoS 130 collects QoS information and sends the QoS information to the device 150 for reconciliation. QoS information may include (but is not limited to) cache usage metrics, memory bandwidth metrics, IO metrics, etc.

[0035] Network adapter silicon hardware 140 collects or includes statistics counters, health status, faults, traffic patterns, port status, etc., and sends this or other information to the device 150.

[0036] The device 150 includes telemetry collector logic and applies heuristics to the collected telemetry data and statistics. The device 150 therefore serves as the local platform detection and enforcement point of SLA, fault management and high availability mediation. Heuristics described with respect to various embodiments can be related to filtering. For example, telemetry can be filtered to focus on a particular VM to make decisions about operation of that particular VM.

[0037] The device 150 may recalculate an improved or optimal configuration and sends a reconfiguration command to all or some of the VMs 160. The device 150 or other system may notify a hypervisor 170 that a reconfiguration has occurred. The reconfiguration can include a re-balance. For example, the PQoS 130 may collect QoS data so that the device 150 can go back to the IA cores to notify the IA cores that a particular VM 160 is using too many resources, so that VM 160 can be assigned to run on a different core.

[0038] Embodiments can provide sleep states and statistics gatherings regarding NICs. It will be appreciated that each port on a NIC has up to 1000 queues that can be associated with a processing element, and the queues can be associated with an application. If one or more of the queues are running low on packets, decisions can be made such as putting the corresponding NIC or a group of NICs to sleep for a certain amount of time until the queues fill up, after which one more NICs will be woken up to continue processing. Embodiments remove the burden of getting such NIC statistics from an IA core.

[0039] Analysis can also be time-oriented such that operators can examine workloads and configurations for optimizations over time. A central controller, for example at a data center level, can perform such filtering and time-based analysis to detect errors and unusual trends.

[0040] FIG. 1 depicts, therefore, a number of platform elements as well as a closed loop system for monitoring hardware and software metrics and making decisions and reconfigurations each of the VMs 160 to reach an improved or optimal platform 100 state.

[0041] FIG. 2 illustrates the device 150 including telemetry collection logic for providing out-of-band telemetry in accordance with some embodiments.

[0042] The device 150 includes at least one telemetry interface 210 to a telemetry collection system. For example, the at least one telemetry interface 210 can interface with the Telemetry Collection Engine 110 for collecting statistics as described earlier herein. The at least one telemetry interface 210 can implement PECI 120 or another protocol. The device 150 can further include at least one platform interface (also incorporated in element 210) to a platform metrics collection system. As described earlier herein, the processing circuitry can gather PQoS metrics over the at least one platform interface 210, and use the PQoS metrics as inputs to the heuristic algorithm. The processing circuitry 200 can determine, based on the heuristic algorithm, whether SLA criteria have been met, and report SLA violations to datacenter management software if SLA criteria have not been met according to decisions or algorithms described earlier herein. The device 150 can include at least one network interface 204 to network adapter hardware 206.

[0043] The device 150 includes processing circuitry 200 configured to receive platform telemetry metrics from the telemetry collection system and network adapter silicon hardware statistics over the at least one network interface 204, to gather collected statistics. In embodiments, the platform telemetry metrics include metrics of at least two metric types selected from a group including processing core data, chipset data, memory element performance data, data received from an encryption unit, data received from a compression unit, storage data, virtual switch (vSwitch) data, and data received over a network interface card (NIC) connection. However, any metrics described earlier herein, or

specified by ETSI NFV or other networking standard or datacenter standard, can be provided to or used by the processing circuitry 200.

[0044] The processing circuitry 200 can apply a heuristic algorithm as described earlier herein using the collected statistics to determine processing core workloads generated by operation of a plurality of VMs 160 communicatively coupled to the device 150.

[0045] The processing circuitry 200 can provide a reconfiguration message as described earlier herein to instruct at least one VM 160 to switch operations to a different processing core, responsive to detecting an overload state on at least one processing core, based on the processing core workloads. In some embodiments, the processing circuitry 200 is configured to provide the reconfiguration message within a request to a hypervisor 170.

[0046] FIG. 3 is a flow diagram of an initializing and benchmarking phase 300 of a NIC affinization algorithm in accordance with some embodiments. FIG. 4 is a flow diagram of an operational phase 400 of a NIC affinization algorithm in accordance with some embodiments. The processing circuitry 200 (FIG. 2) can perform any or all of the operations shown in FIG. 3 and 4, although other elements of the platform 100 (FIG. 1) can also execute some or all of these operations. In some embodiments, the device 150 or the processing circuitry 200 can instruct other elements of the platform 100 in performing any or all operations described with reference to FIGs. 3 and 4.

[0047] Referring to FIG. 3, in operation 304, the processing circuitry 200 can select a core (e.g., IA core) and a type of benchmarking operation to execute. The benchmarking operations can include benchmarking or other evaluations of core-to-cache bandwidth, core-to-I/O bandwidth, core-to-memory bandwidth, or other benchmarks of interest in determining NIC configurations, VM configurations, etc. To perform benchmarking, the processing circuitry 200 will instruct a set of at least two processing cores (e.g., processing cores to be benchmarked), in sequence, to enter an offline state. The processing circuitry 200 will provide instructions for performing tests on each of the set of at least two processing cores after a respective one of the set of at least two processing cores has entered the offline state. In operation 306, the processing circuitry 200 will rank the set of at least two processing cores based on performance during

the benchmarking operations. Subsequent to performing tests, the processing circuitry 200 will generate a ranked set of processing cores. Results of the rankings and tests can be stored in a database or other storage, at a remote or local datacenter central location or other location, or locally to the device 150, or some combination thereof. The method 300 terminates with operation 310, but can be repeated at any time in operation 308.

[0048] Referring to FIG. 4, at operation 402, traffic can be received at a NIC of the platform 100. At operation 404, if an incoming flow is high-priority, associated NIC interrupts can be steered to a high-performance core at operation 406 (as determined based on the rankings generated and stored as described earlier herein). Otherwise, if the incoming flow is not high-priority, associated NIC interrupts can be sent to low-performance cores.

[0049] Embodiments implementing methods 300 and 400 or other similar methods can provide for dynamic detect nonuniformity in shared platform resources (for instance, in some platform embodiments certain cores may have higher available memory bandwidth, and others may have higher I/O bandwidth). The NIC driver could them be affinitized to the highest-performance core(s) and/or highest memory/cache/IO bandwidth core(s) to enhance or improve performance. By determining which cores are best suited to run the NIC drivers for certain hardware devices, embodiments provide better scale-up (within a node), better consolidation and workload density, and potentially improved system-level or workload-level metrics such as higher throughput, reduced jitter or reduced latency.

[0050] The processing circuitry 200 can also be used for performing other functionalities described below with respect to FIGs. 5-10.

[0051] FIG. 5 illustrates NFV system architecture 500 and data flows in accordance with some embodiments. As shown in FIG. 5, the device 150 collects I/O 502, switch 504, and virtual/physical functions 506 telemetry securely. Telemetry is delivered via an OOB network 508 to a NFV Cloud OS agent (e.g., a telemetry agent such as Ceilometer) 510. The telemetry is delivered to the VNF Manager (e.g., Management Console system for Cisco PDN Gateway or Juniper IDS/IPS) 512, which determines the health of the underlying NFV infrastructure (NFVI) according to the requirements of that

VNF (e.g. Cisco PDN Gateway or Juniper IDS/IPS).

[0052] If the NFVI telemetry is deemed problematic (e.g., if there are too many errors, dropped packets, network-based threat in progress, denial of service (DoS) attacks, per- flow/per-tenant/temporal traffic variances, etc.) or if the VNF infrastructure (VNFI) is not meeting the ETSI NFV defined Service Quality metrics defined in accordance with a standard of the ETSI NFV family of standards or a similar standard, then such a situation may be reported to, for example, an orchestrator 514 or other system.

[0053] In addition to telemetry, the device 150 will also enable audits, alarms and controls, as mechanisms for providing SLAs and legal proof of adherence to established SLAs. The device 150 (e.g., an OOB Intel ® IE or ME) will deliver the various Service Quality Metrics requirements, including faults, failures, alarms, and operational misconfigurations, etc., defined by the operators in this spec, to the hypervisor 170, OS, or Cloud OS. Service Quality Metrics include, but not limited to: first-in-first-out (FIFO) depth, flow control events, missed packet count, host buffer or descriptor utilization, Transmission Control Protocol (TCP) congestion window changes, inline Internet Protocol Security (IPsec) or Secure Sockets Layer (SSL) processing metrics and security policies such as checking traffic patterns with the security policy, key lifetime checks, OOB key management, etc. Metrics can also include performance to SLAs, bandwidth, latency, jitter, etc. Metrics can include platform–level metrics such as current cache occupancy, memory bandwidth use, I/O use, etc., by each VM, application, or thread.

[0054] Multiple instantiations 516, 518 of any of the above systems can provide or receive data flows, as shown in FIG. 5. Embodiments are not limited to the exact components or number of components shown in FIG. 5.

[0055] In embodiments, an NFV Manager can be incorporated in the Orchestrator 514 and can take remediation action on the received telemetry (service quality metrics), if the VNF or NFVI are not performing as desired by service quality. In such cases, the VNF Manager 512 can communicate with the Orchestrator 514 for remedial action. The Orchestrator 514 can direct the VIM Workload VNF Life Cycle Management Agent (e.g., enhanced OpenStack Nova) 520 to find an appropriate platform for the VNF. The VNF Life Cycle

Management Agent 520 can perform a remedial action (e.g. VNF Live Migration from existing platform P1 to a new Platform P2, which can meet the expectations of the VNF and VNF manager. The selection of the new platform P2 can be performed by the VNF Life Cycle Management Agent 520 based on the received set of parameters from the VNF Manager 512 (e.g., VNF Descriptor) and the available resources on the potential platforms.

[0056] OOB telemetry can include, by way of non-limiting example: number of NICs, vendor and model for each NIC, type of Peripheral Component Interconnect Express (PCIe) device for each NIC, number of lanes or ports for each NIC, packets per second, packet size, and other packet parameters, PCI Device ID for each port of each NIC, type and size of each port, etc. Regarding VMs, telemetry can include whether each NIC is eligible to be used by each VM, whether each NIC is to be dedicated or shared among VMs, etc. If the NIC is to be shared, telemetry can include whether the NIC is to be shared with sing root I/O virtualization (SR-IOV) or shared through a vSwitch. If shared through SR-IOV, OOB telemetry can include the number of configured virtual functions, a PCI Device ID for each VF, bandwidth or pacing for each VF, etc. If shared through vSwitch, OOB telemetry can include whether a corresponding vSwitch is in bridge mode or network address translation (NAT) mode, number of virtual interfaces, etc. OOB telemetry can include configurations of the supported and disabled functions, offloaded aspects of a NIC or switch function, offload or hardware acceleration policy per tenant, per flow, per SL owner, etc., offload errors, alarms, audits, etc. OOB telemetry can include bandwidth between non-uniform memory access (NUMA) nodes, including total bandwidth and used bandwidth. However, the OOB telemetry examples listed herein are not to be taken as limiting embodiments to any particular OOB telemetry.

OOB Platform Tuning, Configuration and Optimization

[0057] As briefly mentioned earlier herein, embodiments also provide for improved synchronization and accurate telemetry across servers within racks, across the datacenter, and across multiple datacenters spanning multiple geographical locations. Such synchronization is an issue in datacenter operations that inhibits synchronized delivery of cloud services, ensuring that the

user always observes the latest copies of data. Embodiments provide improved tuning performance of platform and workloads and tracking that behavior over time and share resource reallocation to meet SLA targets.

[0058] Some synchronizations methods use in-band methods, which require OS/VMM involvement on each system involved. Embodiments provide an OOB approach, described herein.

[0059] FIG. 6 is a block diagram of a system 600 for OOB platform configuration parameter configurability in accordance with some embodiments.

[0060] As shown in FIG. 6, an independent hardware/firmware agent 602 communicates with a management and policy server 604 to send/receive information including configuration inputs 606 and performance feedback 608. This hardware/firmware agent 602 may communicate with the management and policy server 604 over a standard network link or over a specialized network with lower congestion to lower latency.

[0061] The hardware/firmware agent 602 may communicate with the rest of the platform 610 (shared resources, OS/VMM, application software, etc.) via other interfaces or protocols, which may include shared memory regions (mailbox-style approaches) or interrupts.

[0062] The hardware/firmware agent 602 may be implemented in Intel ® IE (customer customizable) or Intel ® ME (closed-source) although embodiments are not limited thereto. Components of the device 150 (FIG. 1) may also be included in the hardware/firmware agent 602.

[0063] Since the hardware/firmware agent 602 operates independently of the rest of the platform 600, the hardware/firmware agent 602 can asynchronously receive configuration inputs from the management and policy server 604 in the datacenter (which in turn may have received policy updates from another datacenter in a geographically different region). Accordingly, the hardware/firmware agent 602 can apply these updates to the platform 600 after processing, wherein processing includes configuration checking, merge, applicability filtering, parameter modification, etc. The hardware/firmware agent 602 may also communicate with many other platform elements 610 such as OS/VMM software, individual applications, performance monitoring counters, or shared resources such as the L3 cache and memory to measure

sharing of such shared resources on a per-application basis. These metrics can then be provided back to the management and policy server 604 in order to guide advanced resource-aware scheduling decisions, to meet SLAs or other average performance targets, or to provide metrics to the datacenter administrator to measure metrics such as cache and memory contention and bandwidth utilization, aggregated or reported per-platform.

[0064] In addition to configuration changes, embodiments can provide platform optimizations, such as tuning prefetchers, in real time based on the workloads that are running, in order to provide higher performance. Such fine-grained tuning algorithms may be run either at the management and policy server 604 or at hardware/firmware agent 602 depending on datacenter goals and the level of logging and visibility required.

[0065] The asynchronous hardware/firmware agent 602 and its interfaces to the management and policy server 604 and the rest of the platform 600, including hardware and software, provide a set of OOB capabilities as described herein. The hardware/firmware agent 602 can include compute resources consisting of one of more cores, memory resources, telemetry data and other data, a configuration state passed down from the management and policy server 604, which may be modified locally before applying to the system, and performance data read back from the platform 600. Algorithms running in the hardware/firmware agent 602 or a core therein can act upon performance feedback data and node/workload mappings and policies (which may include performance targets) to determine whether performance targets are met. These algorithms may include simple policies to maximize a single variable (such as system throughput or the performance of a specific workload) or more complex (e.g., involving multiple platform inputs and multivariate parameter maximization or optimization schemes, or complex algorithms to compare performance of multiple workloads to individual performance targets). These algorithms can act upon input performance data to make reconfiguration decisions to provide to the reset of the platform 600. These reconfiguration changes may change the behavior of the platform 600, thereby modifying the performance metrics reported back to the hardware/firmware agent 602, thereby forming a closed-loop control system consisting of the hardware/firmware agent

602, the management and policy server 604, the performance feedback, and the rest of the platform. The management and policy server 604 can be centralized or distributed in various embodiments.

[0066] The management and policy server 604 can include a state table or similar tracking system that tracks per-node state of workloads, policies, cache sensitivity, bandwidth sensitivity and other pertinent workload sensitivity data and performance metrics.

[0067] The hardware/firmware agent 602 provides performance monitoring data to the management and policy server 604. The performance monitoring data may be sampled from a variety of sources, including application feedback, OS feedback, or hardware sources, such as performance monitoring counters and resource monitoring counters. The performance monitoring data can provide detailed information on L3 cache utilization, memory bandwidth utilization, I/O bandwidth utilization, etc. These sources of information can be cleaned and optionally averaged and/or compressed before sending to the management and policy server 604, which maintains this information by mapping node and workload to each of the parameters and running algorithms on top of this data to determine optimal configuration settings. The management and policy server 604 may maintain a table or database mapping workloads, nodes, and performance characteristics to aid decision making and tracking of application characteristics across time.

[0068] The management and policy server 604 can push changes of timing data or configuration state to each server, or to other datacenters. Examples may include using these OOB mechanisms for time synchronization or for pushing configuration changes to switch to a more power-efficient operating mode for some servers during low-load times. These updates may be pushed over a standard or dedicated network interface to each platform (depending on datacenter network topology).

[0069] Once the hardware/firmware agent 602 receives a configuration update request from management and policy server 604, the hardware/firmware agent 602 can perform basic checking (e.g., checking whether the requested value is within a safe range, whether the requested configuration parameter is supported on this platform, etc.). The hardware/firmware agent 602 either can buffer the

change to apply at a preset time (the present time may be specified with the message) or the hardware/firmware agent can apply the request immediately or as immediately as technologically feasible given network conditions, etc.

[0070] The hardware/firmware agent 602 can update parameters such as prefetchers settings, data directed I/O (DDIO), RDT allocation settings, PQoS, C-state settings, P-state settings (e.g.., SpeedStep), OS configuration settings, application configuration settings or other configuration parameters request by the management and policy server 604.

[0071] The hardware/firmware agent 602 may also independently run algorithms to tune system state, modulating the parameters previously listed or others. Performance aggregation algorithms, and an evaluation of the effectiveness thereof, is provided below with respect to FIGs. 7-8. FIG. 7 illustrates performance versus cache occupancy of a cache-sensitive workload. FIG. 8 illustrates performance versus cache sensitivity of a compute-bound workload that does not exhibit sensitivity to cache resources.

[0072] A plot similar to that shown in FIG. 7 can be generated by running a cache-sensitive application in the presence of many other applications including cache-intensive, compute-intensive, and memory-intensive applications, on the platform 600 (FIG. 6) or similar platform. FIG. 7 illustrates a detailed and accurate view of performance vs. cache occupancy (e.g., cache sensitivity). Embodiments can build such sensitivity curves to enable scheduling for single workloads on a server, as well as for all workloads in a datacenter simultaneously.

[0073] FIG. 8 illustrates another example plot as can be generated by running a compute-sensitive workload in the presence of many other applications including cache-intensive, compute-intensive, and memory-intensive applications, on the platform 600 (FIG. 6) or similar platform as can be seen in a typical datacenter. As will be appreciated, the compute-sensitive workload does not show sensitivity to shared resources such as last-level cache. Embodiments can detect and track such workloads at a fine-grained level in a dynamic datacenter.

[0074] FIG. 9 is a flow diagram of an example hardware-implemented method 900 for implementing a performance monitoring and aggregation algorithm in

accordance with some embodiments. The device 150 (FIG. 1), the hardware/firmware agent 602, or another device or apparatus can perform one or more operations of example hardware-implemented method 900. According, the hardware/firmware agent 602 can execute performance monitoring aggregation algorithms in various embodiments to profile applications, as one part of a multi-faceted set of profiling algorithms.

[0075] The example method 900 begins with operation 902 with the hardware/firmware agent 602 assigning a resource monitoring identifier (RMID) to each thread of the application. The hardware/firmware agent 602 may use a technology such as Intel Cache Monitoring Technology (CMT) or Intel Memory Bandwidth Monitoring (MBM) in operation 902, although embodiments are not limited thereto.

[0076] The example method 900 continues with operation 904 with the hardware/firmware agent 602 associating an RMID with a hardware thread. In some embodiments, the hardware/firmware agent 602 may perform operation 904 on context swaps onto a core. In operation 904, therefore, software is instructing hardware to monitor the thread, which can be more computationally efficient relative to software thread monitoring. Software can later retrieve metrics such as instructions per cycle, etc.

[0077] The example method 900 continues with operation 906 with the hardware/firmware agent 602 periodically sampling the performance monitoring event codes for cache occupancy and memory bandwidth (via the IA32_QM_EVTSEL and IA32_QM_CTR MSR interfaces, for example), and sampling the performance of the application (via instructions per cycle (IPC), application-reported performance such as transactions per second, etc.).

[0078] The example method 900 continues with operation 908 with creation of performance predictions. In executing operation 908, the hardware/firmware agent 602 can store values retrieved in memory to build a history over time. After a period ranging from seconds to days, the hardware/firmware agent 602 can process the data by "bucketing" into categories of cache occupancy (e.g., 0-1 MB, 1-2 MB, 2-3 MB, etc. as buckets for cache occupancy) and average the performance values for each "bucket." The hardware/firmware agent 602 can fit a curve to the given points, creating a fit for memory bandwidth or cache

occupancy vs. performance.

[0079] The hardware/firmware agent 602 or other system can check a correlation coefficient to confirm that the correlation coefficient is sufficiently high to provide usable and accurate performance predictions for cache occupancy or memory bandwidth inputs. The coefficients can be saved in tables described earlier herein with reference to FIG. 6 or in other memory.

[0080] The hardware/firmware agent 602 may take derivatives of the curves to create curves that model performance sensitivity vs. cache occupancy or performance bandwidth. The hardware/firmware agent 602 or other component of the platform 600 (FIG. 6) can use these models to make threshold-based decisions as to how much cache or memory bandwidth an application actually needs.

[0081] For example, with reference to FIG. 10, the optimal cache operating point of an application can be defined as the point A where application performance improves less than 2% (or some other threshold amount or percentage) by providing an additional 1 MB of L3 cache). FIG. 10 illustrates cache sensitivity data for a cache-sensitive workload that can be analyzed for making configuration decisions in accordance with some embodiments. FIG. 10 was formed for a cache-sensitive workload, by taking the derivative of a curve fit to the original data. In embodiments, one or more components of the platform 600 or other component or computing system described herein, can include a display for displaying curves of FIG. 10, or any other curve produced in the course of providing analysis of cache sensitivity, cache operating points, etc.

[0082] Referring again to FIG. 9, any or all of the operations of example method 900 can be repeated periodically to either rebuild or augment the performance prediction curves on a per-app/thread/VM basis. Analyses such as those described above can allow advanced workload placement decisions to be made in real time. For instance if a workload is found to be cache-sensitive, and is specified to be high-priority by the datacenter administrator, that workload could be moved to a server with low cache utilization for better performance. Alternately, using systems and apparatuses in accordance with

various embodiments, the central controller/datacenter manager could push an update to the server to force the system to reconfigure the caches to reserve a larger portion of the cache for this cache-sensitive workload. These types of updates are possible in real-time, without the need for the datacenter administrator to intervene thanks to the closed- loop software control provided in various embodiments.

[0083] Though the example embodiments described above are based on a datacenter environment, which may be running bare metal or virtualized workloads, OOB platform monitoring and configuration in accordance with various embodiments is applicable across multiple scenarios, including communication workloads, and NFV/SDN scenarios, where the priority of certain flows is updated in real-time with low latency, for instance.

[0084] The example method 900 can include any other operations or functionalities of a device 150, a hardware/firmware agent 602, or usage model thereof, described above with respect to FIGs. 1-8. Operations can be performed in any order or in parallel where appropriate. The method 900 can be performed by hardware, firmware, software, or any combination thereof.

[0085] For example, in some embodiments, the example method 900 can include processing circuitry 200 (FIG. 2) or other elements receiving a configuration state from a management and policy server 604, the configuration state including at least one processing core identifier and at least one of a workload, a policy, a cache sensitivity, and a bandwidth sensitivity for the respective at least one processing core identifier; providing performance feedback, to the management and policy server, for at least one processing core identified by the at least one processing core identifier; and receiving recommendations from the management and policy server for providing the reconfiguration message, based on the performance feedback. Upon receiving performance monitoring event codes corresponding to a parameter of interest, the processing circuitry 200 or other component can detect application performance to generate a performance curve relating application performance to the parameter of interest; generate a sensitivity curve, from the performance curve, to determine sensitivity of application performance to the parameter of interest; and provide the sensitivity curve as an input to an algorithm for

generating reconfiguration decisions. The parameter of interest can include one of cache occupancy and memory bandwidth.

[0086] Examples, as described herein, may include, or may operate on, logic or a number of components, modules, or mechanisms. Modules are tangible entities (e.g., hardware) capable of performing specified operations and may be configured or arranged in a certain manner. In an example, circuits may be arranged (e.g., internally or with respect to external entities such as other circuits) in a specified manner as a module. In an example, at least a part of one or more computer systems (e.g., a standalone, client or server computer system) or one or more processors of the device 150 or the hardware/firmware agent 602 may be configured by firmware or software (e.g., instructions 202 (FIG. 2), an application portion, or an application) as a module that operates to perform specified operations. In an example, the software may reside on at least one machine-readable medium. In an example, the software, when executed by the underlying hardware of the module (e.g., the device 150 or the hardware/firmware agent 602), can include instructions 202 (FIG. 2) to cause the hardware to perform the specified operations.

[0087] For example, instructions 202 can cause hardware to receive periodically over a time duration, performance monitoring event codes related to at least one of memory bandwidth and cache occupancy for a computing platform. The instructions 202 can cause the hardware to periodically detect application performance for an application executing on the computing platform, responsive to periodically receiving the performance monitoring event codes, to generate at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for the computing platform.

[0088] In various embodiments, the instructions 202 can cause the hardware to determine sensitivity of application performance to at least one of memory bandwidth and cache occupancy based on a first derivative of the at least one curve. The instructions 202 can cause the hardware to generate a configuration decision for the computing platform based on sensitivity of application performance to at least one of memory bandwidth and cache occupancy.

[0089] In some embodiments, the instructions 202 can cause the hardware to assign a resource monitoring identifier (RMID) to each thread of an application

and analyzing one of instructions per cycle and transactions per second of application threads based on respective RMIDs.

[0090] The term "module" is understood to encompass a tangible entity, be that an entity that is physically constructed, specifically configured (e.g., hardwired), or temporarily (e.g., transitorily) configured (e.g., programmed) to operate in a specified manner or to perform at least part of any operation described herein. Considering examples in which modules are temporarily configured, a module need not be instantiated at any one moment in time. For example, where the modules comprise a general-purpose hardware processor configured using software; the general-purpose hardware processor may be configured as respective different modules at different times. Software may accordingly configure a hardware processor, for example, to constitute a particular module at one instance of time and to constitute a different module at a different instance of time. The term "application," or variants thereof, is used expansively herein to include routines, program modules, programs, components, and the like, and may be implemented on various system configurations, including single-processor or multiprocessor systems, microprocessor-based electronics, single-core or multi-core systems, combinations thereof, and the like. Thus, the term application may be used to refer to an embodiment of software or to hardware arranged to perform at least part of any operation described herein.

[0091] While a machine-readable medium may include a single medium, the term "machine-readable medium" may include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers).

[0092] The term "machine-readable medium" may include any medium that is capable of storing, encoding, or carrying instructions 202 for execution by a machine (e.g., the device 150 or any other module) and that cause the machine to perform any one or more of the techniques of the present disclosure, or that is capable of storing, encoding or carrying data structures used by or associated with such instructions. In other words, the processing circuitry 200 (FIG. 2) can include instructions and can therefore be termed a machine-readable medium in the context of various embodiments. Other non-limiting machine-readable

medium examples may include solid-state memories, and optical and magnetic media. Specific examples of machine-readable media may include: non-volatile memory, such as semiconductor memory devices (e.g., Electrically Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM)) and flash memory devices; magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks.

[0093] The instructions 202 may further be transmitted or received over a communications network using a transmission medium utilizing any one of a number of transfer protocols (e.g., frame relay, internet protocol (IP), TCP, user datagram protocol (UDP), hypertext transfer protocol (HTTP), etc.). Example communication networks may include a local area network (LAN), a wide area network (WAN), a packet data network (e.g., the Internet), mobile telephone networks ((e.g., channel access methods including Code Division Multiple Access (CDMA), Time-division multiple access (TDMA), Frequency-division multiple access (FDMA), and Orthogonal Frequency Division Multiple Access (OFDMA) and cellular networks such as Global System for Mobile Communications (GSM), Universal Mobile Telecommunications System (UMTS), CDMA 2000 1x* standards and Long Term Evolution (LTE)), Plain Old Telephone (POTS) networks, and wireless data networks (e.g., Institute of Electrical and Electronics Engineers (IEEE) 802 family of standards including IEEE 802.11 standards (WiFi), IEEE 802.16 standards (WiMax®) and others), peer-to-peer (P2P) networks, or other protocols now known or later developed.

[0094] The term "transmission medium" shall be taken to include any intangible medium that is capable of storing, encoding or carrying instructions for execution by hardware processing circuitry, and includes digital or analog communications signals or other intangible medium to facilitate communication of such software.

Additional Notes & Examples:

[0095] Example 1 includes subject matter (such as a control device, interplane control device, control plane processor, computer device and or any other electrical apparatus, device or processor) including at least one telemetry

interface to a telemetry collection system; at least one network interface to network adapter hardware; and processing circuitry configured to receive platform telemetry metrics from the telemetry collection system, and network adapter silicon hardware statistics over the at least one network interface, to gather collected statistics, apply a heuristic algorithm using the collected statistics to determine processing core workloads generated by operation of a plurality of software systems communicatively coupled to the device, and provide a reconfiguration message to instruct at least one software system to switch operations to a different processing core, responsive to detecting an overload state on at least one processing core, based on the processing core workloads.

[0096] In Example 2, the subject matter of Example 1 can optionally include wherein the plurality of software systems includes at least one virtual machine (VM).

[0097] In Example 3, the subject matter of any of Examples 1-2 can optionally include wherein the processing circuitry is configured to provide the reconfiguration message within a request to a hypervisor.

[0098] In Example 4, the subject matter of any of Examples 1-3 can optionally include wherein the platform telemetry metrics include metrics of at least two metric types selected from a group including processing core data, chipset data, memory element performance data, data received from an encryption unit, data received from a compression unit, storage data, virtual switch (vSwitch) data, and data received over a network interface card (NIC) connection, wherein data received over the NIC includes NIC telemetry, wherein NIC telemetry includes at least one of an indication of packets per second received at the NIC and average packet size received at the NIC.

[0099] In Example 5, the subject matter of any of Examples 1-4 can optionally include at least one platform interface to a platform metrics collection system, and wherein the processing circuitry is further configured to gather platform quality of service (PQoS) metrics over the at least one platform interface, and to use the PQoS metrics as inputs to the heuristic algorithm.

[00100] In Example 6, the subject matter of any of Examples 1-5 can optionally include wherein the processing circuitry is further configured to instruct a set of

at least two processing cores, in sequence, to enter an offline state; provide instructions for performing tests on each of the set of at least two processing cores after a respective one of the set of at least two processing cores has entered the offline state; and rank the set of at least two processing cores based on performance during the tests, subsequent to performing tests, to generate a ranked set of processing cores.

[00101] In Example 7, the subject matter of Example 6 can optionally include wherein the tests include evaluations of at least one of core-to-cache bandwidth, core-to-memory bandwidth, and core-to-I/O bandwidth.

[00102] In Example 8, the subject matter of any of Examples 6-7 can optionally include wherein the processing circuitry is further configured to provide instructions for steering incoming NIC traffic to a processing core of the ranked set of processing cores, based on priority level of the incoming NIC traffic.

[00103] In Example 9, the subject matter of any of Examples 1-8 can optionally include herein the processing circuitry is further arranged to determine, based on the heuristic algorithm, whether service level agreement (SLA) criteria have been met; and report SLA violations to datacenter management software if SLA criteria have not been met.

[00104] In Example 10, the subject matter of any of Examples 1-9 can optionally include wherein the processing circuitry is further arranged to receive a configuration state from a management and policy server, the configuration state including at least one processing core identifier and at least one of a workload, a policy, a cache sensitivity, and a bandwidth sensitivity for the respective at least one processing core identifier; provide performance feedback, to the management and policy server, for at least one processing core identified by the at least one processing core identifier; and receive recommendations from the management and policy server for providing the reconfiguration message, based on the performance feedback.

[00105] In Example 11, the subject matter of Example 10 can optionally include wherein the processing circuitry is further arranged to upon receiving performance monitoring event codes corresponding to a parameter of interest, detect application performance to generate a performance curve relating application performance to the parameter of interest; generate a sensitivity curve,

from the performance curve, to determine sensitivity of application performance to the parameter of interest; and provide the sensitivity curve as an input to an algorithm for generating reconfiguration decisions.

[00106] In Example 12, the subject matter of Example 11 can optionally include wherein the parameter of interest includes one of cache occupancy and memory bandwidth, and wherein cache occupancy is independent of memory bandwidth.

[00107] Example 13 includes subject matter such as a machine-readable medium including instructions that, when executed on a machine (such as a control device, interplane control device, Innovation Engine, Management Engine, control plane processor, computing device, NIC card, etc.) cause the machine to receive, periodically over a time duration, performance monitoring event codes related to at least one of memory bandwidth and cache occupancy for a computing platform; periodically detect application performance for an application executing on the computing platform, responsive to periodically receiving the performance monitoring event codes, to generate at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for the computing platform; determine sensitivity of application performance to at least one of memory bandwidth and cache occupancy based on a first derivative of the at least one curve; and generate a configuration decision for the computing platform based on sensitivity of application performance to at least one of memory bandwidth and cache occupancy.

[00108] In Example 14, the subject matter of Example 13 can optionally include further instructions to cause the machine to assign a resource monitoring identifier (RMID) to each thread of the application; and analyze one of instructions per cycle and transactions per second of application threads based on respective RMIDs.

[00109] In Example 15, the subject matter of Example 14 can optionally include further instructions to cause the machine to generate a cache operating point for the application by determining a point, based on application sensitivity curve, at which application performance is improved by less than a threshold amount for an additional unit measurement of cache; and provide a configuration decision to specify that the application should execute on a processing core with low cache utilization if the cache operating point indicates that the application has a high

level of cache sensitivity.

[00110] In Example 16, the subject matter of Example 15 can optionally include further instructions to cause the machine to provide the at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for display on a central management engine.

[00111] Example 17 includes subject matter include a method, the method comprising receiving platform telemetry metrics from a telemetry collection system, and network adapter silicon hardware statistics over at least one network interface, to gather collected statistics; applying a heuristic algorithm using the collected statistics to determine processing core workloads generated by operation of a plurality of virtual machines (VMs) communicatively coupled to the device; and providing a reconfiguration message to a hypervisor to instruct at least one VM associated with the hypervisor to switch operations to a different processing core, responsive to detecting an overload state on at least one processing core, based on the processing core workloads.

[00112] In Example 18, the subject matter of Example 17 can optionally include wherein the platform telemetry metrics include metrics of at least two metric types selected from a group including processing core data, chipset data, memory element performance data, data received from an encryption unit, data received from a compression unit, storage data, virtual switch (vSwitch) data, and data received over a network interface card (NIC) connection.

[00113] In Example 19, the subject matter of any of Examples 17-18 can optionally include instructing a set of at least two processing cores to enter, in sequence, an offline state; providing instructions for performing tests on each of the set of at least two processing cores after a respective one of the set of at least two processing cores has entered the offline state; ranking the set of at least two processing cores based on performance during the tests, subsequent to performing tests, to generate a ranked set of processing cores; and providing instructions for steering incoming network interface card (NIC) traffic to a processing core of the ranked set of processing cores, based on priority level of the incoming NIC traffic.

[00114] In Example 20, the subject matter of Example 19 can optionally include herein the tests include evaluations of at least one of core-to-cache bandwidth,

core-to-memory bandwidth, and core-to-input/output bandwidth.

[00115] In Example 21, the subject matter of any of Examples 17-20 can optionally include receiving, periodically over a time duration, performance monitoring event codes related to at least one of memory bandwidth and cache occupancy for a computing platform that includes the processing cores; periodically detecting application performance for an application executing on the computing platform, responsive to periodically receiving the performance monitoring event codes, to generate at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for the computing platform; determining sensitivity of application performance to at least one of memory bandwidth and cache occupancy based on a first derivative of the at least one curve; and generating a configuration decision for the computing platform based on sensitivity of application performance to at least one of memory bandwidth and cache occupancy.

[00116] The above detailed description includes references to the accompanying drawings, which form a part of the detailed description. The drawings show, by way of illustration, specific embodiments that may be practiced. These embodiments are also referred to herein as "examples." Such examples may include elements in addition to those shown or described. However, also contemplated are examples that include the elements shown or described. Moreover, also contemplate are examples using any combination or permutation of those elements shown or described (or one or more aspects thereof), either with respect to a particular example (or one or more aspects thereof), or with respect to other examples (or one or more aspects thereof) shown or described herein.

[00117] Publications, patents, and patent documents referred to in this document are incorporated by reference herein in their entirety, as though individually incorporated by reference. In the event of inconsistent usages between this document and those documents so incorporated by reference, the usage in the incorporated reference(s) are supplementary to that of this document; for irreconcilable inconsistencies, the usage in this document controls.

[00118] In this document, the terms "a" or "an" are used, as is common in

patent documents, to include one or more than one, independent of any other instances or usages of "at least one" or "one or more." In this document, the term "or" is used to refer to a nonexclusive or, such that "A or B" includes "A but not B," "B but not A," and "A and B," unless otherwise indicated. In the appended claims, the terms "including" and "in which" are used as the plain-English equivalents of the respective terms "comprising" and "wherein." Also, in the following claims, the terms "including" and "comprising" are open-ended, that is, a system, device, article, or process that includes elements in addition to those listed after such a term in a claim are still deemed to fall within the scope of that claim. Moreover, in the following claims, the terms "first," "second," and "third," etc. are used merely as labels, and are not intended to suggest a numerical order for their objects.

[00119] The above description is intended to be illustrative, and not restrictive. For example, the above-described examples (or one or more aspects thereof) may be used in combination with others. Other embodiments may be used, such as by one of ordinary skill in the art upon reviewing the above description. The Abstract is to allow the reader to quickly ascertain the nature of the technical disclosure and is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims. Also, in the above Detailed Description, various features may be grouped together to streamline the disclosure. However, the claims may not set forth features disclosed herein because embodiments may include a subset of said features. Further, embodiments may include fewer features than those disclosed in a particular example. Thus, the following claims are hereby incorporated into the Detailed Description, with a claim standing on its own as a separate embodiment. The scope of the embodiments disclosed herein is to be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

CLAIMS

What is claimed is:

1. A device comprising:

at least one telemetry interface to a telemetry collection system;

at least one network interface to network adapter hardware; and

processing circuitry configured to

receive platform telemetry metrics from the telemetry collection system, and network adapter silicon hardware statistics over the at least one network interface, to gather collected statistics,

apply a heuristic algorithm using the collected statistics to determine processing core workloads generated by operation of a plurality of virtual machines (VMs) communicatively coupled to the device, and

provide a reconfiguration message to instruct at least one VM to switch operations to a different processing core, responsive to detecting an overload state on at least one processing core, based on the processing core workloads.

2. The device of claim 1, wherein the processing circuitry is configured to provide the reconfiguration message within a request to a hypervisor.

3. The device of claim 1, wherein the platform telemetry metrics include metrics of at least two metric types selected from a group including processing core data, chipset data, memory element performance data, data received from an encryption unit, data received from a compression unit, storage data, virtual switch (vSwitch) data, and data received over a network interface card (NIC) connection.

4. The device of claim 1, further comprising:

at least one platform interface to a platform metrics collection system, and wherein

the processing circuitry is further configured to gather platform quality of service (PQoS) metrics over the at least one platform interface, and to use the PQoS metrics as inputs to the heuristic algorithm.

5. The device of claim 1, wherein the processing circuitry is further configured to:

instruct a set of at least two processing cores, in sequence, to enter an offline state;

provide instructions for performing tests on each of the set of at least two processing cores after a respective one of the set of at least two processing cores has entered the offline state; and

rank the set of at least two processing cores based on performance during the tests, subsequent to performing tests, to generate a ranked set of processing cores.

6. The device of claim 5, wherein the tests include evaluations of at least one of:

core-to-cache bandwidth,

core-to-memory bandwidth, and

core-to-I/O bandwidth.

7. The device of claim 6, wherein the processing circuitry is further configured to:

provide instructions for steering incoming NIC traffic to a processing core of the ranked set of processing cores, based on priority level of the incoming NIC traffic.

8. The device of claim 1, wherein the processing circuitry is further arranged to:

determine, based on the heuristic algorithm, whether service level agreement (SLA) criteria have been met; and

report SLA violations to datacenter management software if SLA criteria have not been met.

9. The device of claim 1, wherein the processing circuitry is further arranged to:

receive a configuration state from a management and policy server, the configuration state including at least one processing core identifier and at least one of a workload, a policy, a cache sensitivity, and a bandwidth sensitivity for the respective at least one processing core identifier; and

provide performance feedback, to the management and policy server, for at least one processing core identified by the at least one processing core identifier; and

receive recommendations from the management and policy server for providing the reconfiguration message, based on the performance feedback.

10. The device of claim 9, wherein the processing circuitry is further arranged to:

upon receiving performance monitoring event codes corresponding to a parameter of interest, detect application performance to generate a performance curve relating application performance to the parameter of interest;

generate a sensitivity curve, from the performance curve, to determine sensitivity of application performance to the parameter of interest; and

provide the sensitivity curve as an input to an algorithm for generating reconfiguration decisions.

11. The device of claim 10, wherein the parameter of interest includes one of cache occupancy and memory bandwidth.

12. A machine-readable medium including instructions that, when executed on a machine cause the machine to perform operations including:

receiving, periodically over a time duration, performance monitoring event codes related to at least one of memory bandwidth and cache occupancy for a computing platform;

periodically detecting application performance for an application executing on the computing platform, responsive to periodically receiving the performance monitoring event codes, to generate at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for the computing platform;

determining sensitivity of application performance to at least one of memory bandwidth and cache occupancy based on a first derivative of the at least one curve; and

generating a configuration decision for the computing platform based on sensitivity of application performance to at least one of memory bandwidth and cache occupancy.

13. The machine-readable medium of claim 12, including instructions that, when executed on the machine, cause the machine to detect application performance by performing operations including:

assigning a resource monitoring identifier (RMID) to each thread of the application; and

analyzing one of instructions per cycle and transactions per second of application threads based on respective RMIDs.

14. The machine-readable medium of claim 13, including instructions that, when executed on the machine, cause the machine to perform operations including:

generating a cache operating point for the application by determining a point, based on application sensitivity curve, at which application performance is improved by less than a threshold amount for an additional unit measurement of cache; and

providing a configuration decision to specify that the application should execute on a processing core with low cache utilization if the cache operating point indicates that the application has a high level of cache sensitivity.

15. The machine-readable medium of claim 14, including instructions that, when executed on the machine, cause the machine to perform operations including:

providing the at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for display on a central management engine.

16. A method for platform processing core configuration, the method comprising:

receiving platform telemetry metrics from a telemetry collection system, and network adapter silicon hardware statistics over at least one network interface, to gather collected statistics;

applying a heuristic algorithm using the collected statistics to determine processing core workloads generated by operation of a plurality of virtual machines (VMs) communicatively coupled to the device; and

provide a reconfiguration message to a hypervisor to instruct at least one VM associated with the hypervisor to switch operations to a different processing core, responsive to detecting an overload state on at least one processing core, based on the processing core workloads.

17. The method of claim 16, wherein the platform telemetry metrics include metrics of at least two metric types selected from a group including processing core data, chipset data, memory element performance data, data received from an encryption unit, data received from a compression unit, storage data, virtual switch (vSwitch) data, and data received over a network interface card (NIC) connection.

18. The method of claim 16, further comprising:

instructing a set of at least two processing cores to enter, in sequence, an offline state;

providing instructions for performing tests on each of the set of at least two processing cores after a respective one of the set of at least two processing cores has entered the offline state;

ranking the set of at least two processing cores based on performance during the tests, subsequent to performing tests, to generate a ranked set of processing cores; and

providing instructions for steering incoming network interface card (NIC) traffic to a processing core of the ranked set of processing cores, based on priority level of the incoming NIC traffic.

19. The method of claim 18, wherein the tests include evaluations of at least one of:

core-to-cache bandwidth,

core-to-memory bandwidth, and

core-to-input/output bandwidth.

20. The method of claim 16, further comprising:

receiving, periodically over a time duration, performance monitoring event codes related to at least one of memory bandwidth and cache occupancy for a computing platform that includes the processing cores;

periodically detecting application performance for an application executing on the computing platform, responsive to periodically receiving the performance monitoring event codes, to generate at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for the computing platform;

determining sensitivity of application performance to at least one of memory bandwidth and cache occupancy based on a first derivative of the at least one curve; and

generating a configuration decision for the computing platform based on sensitivity of application performance to at least one of memory bandwidth and cache occupancy.

21. An apparatus comprising means for performing any of the methods of claims 16-20.

22. A device comprising:

a networking interface to receive, periodically over a time duration, performance monitoring event codes related to at least one of memory bandwidth and cache occupancy for a computing platform; and

processing circuitry configured to:

periodically detect application performance for an application executing on the computing platform, responsive to periodically receiving the performance monitoring event codes, to generate at least one curve relating application performance to at least one of memory bandwidth and cache occupancy for the computing platform;

determine sensitivity of application performance to at least one of memory bandwidth and cache occupancy based on a first derivative of the at least one curve; and

generate a configuration decision for the computing platform based on sensitivity of application performance to at least one of memory bandwidth and cache occupancy.

23. The device of claim 22, wherein the processing circuitry is further configured to:

assign a resource monitoring identifier (RMID) to each thread of the application; and

analyze one of instructions per cycle and transactions per second of application threads based on respective RMIDs.

24. The device of claim 23, wherein the processing circuitry is further configured to:

generate a cache operating point for the application by determining a point, based on application sensitivity curve, at which application performance is improved by less than a threshold amount for an additional unit measurement of cache; and

provide a configuration decision to specify that the application should execute on a processing core with low cache utilization if the cache operating point indicates that the application has a high level of cache sensitivity.

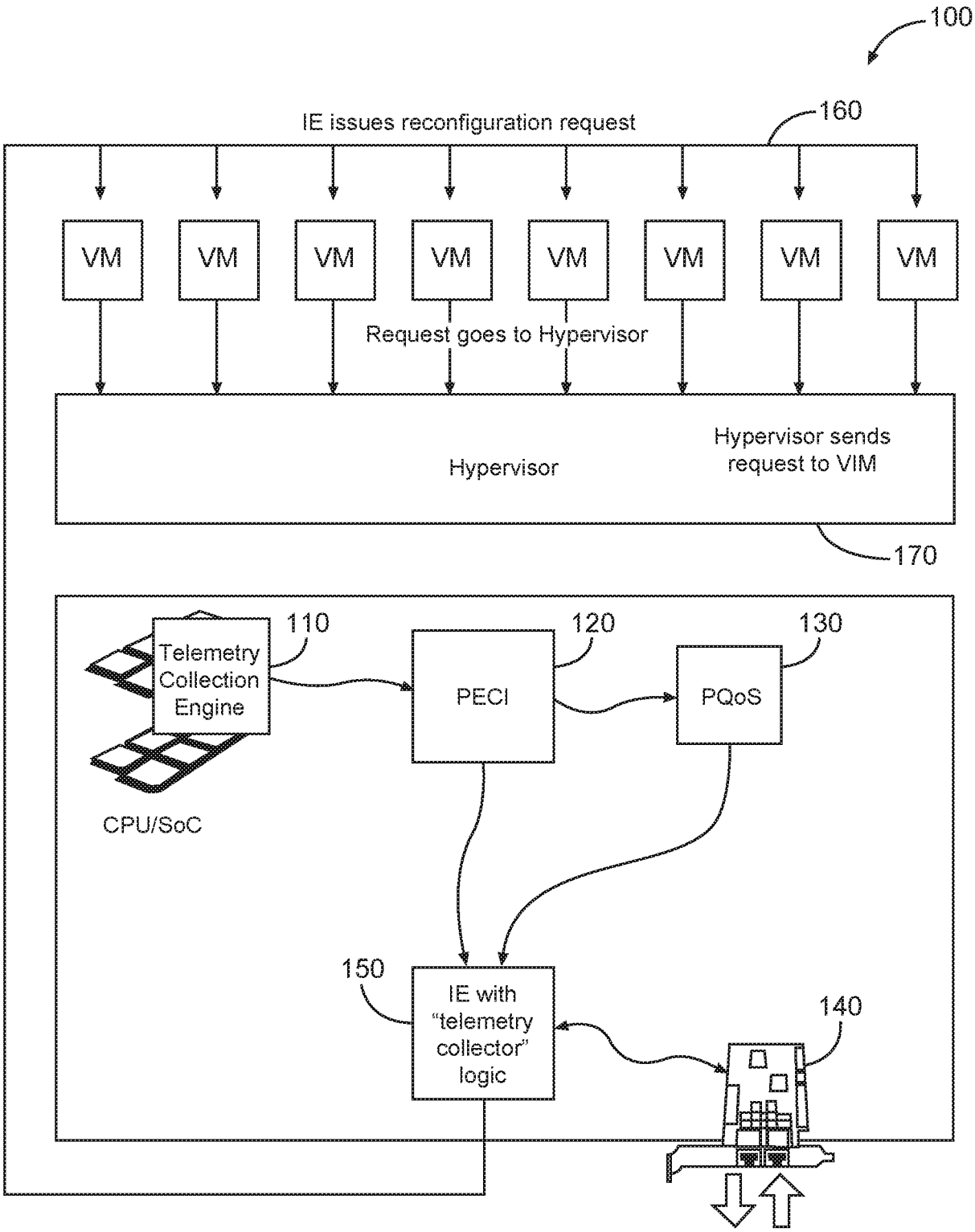
100
IE issues reconfiguration request
160
VM
VM
VM
VM
VM
VM
VM
VM
Request goes to Hypervisor
Hypervisor
Hypervisor sends request to VIM
170
110
120
130
Telemetry Collection Engine
PECI
PQoS
CPU/SoC
150
IE with "telemetry collector" logic
140

Fig. 1

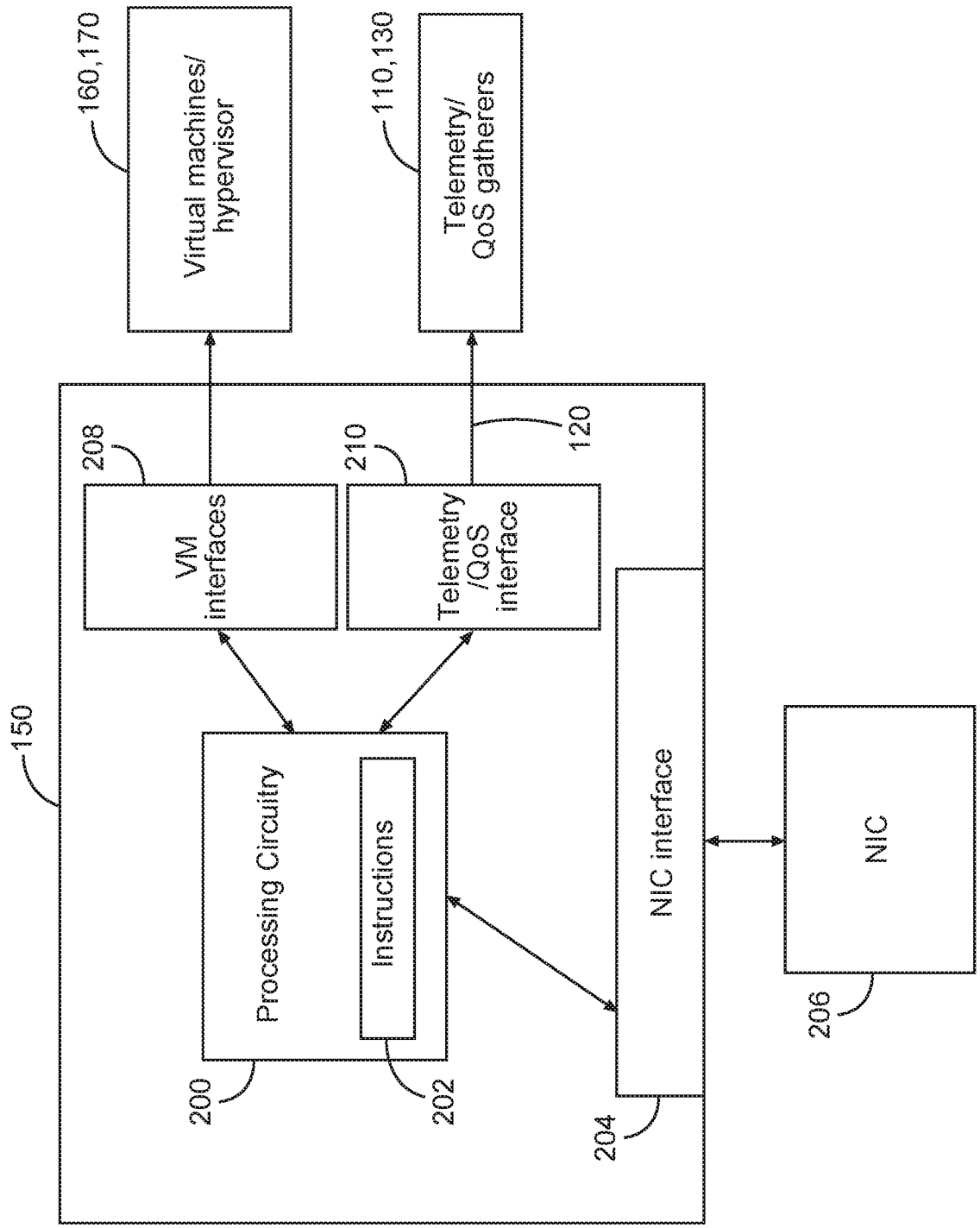
160,170
Virtual machines/ hypervisor
110,130
Telemetry/ QoS gatherers
208
VM interfaces
210
Telemetry /QoS interface
120
150
Processing Circuitry
Instructions
200
202
NIC interface
204
NIC
206

Fig. 2

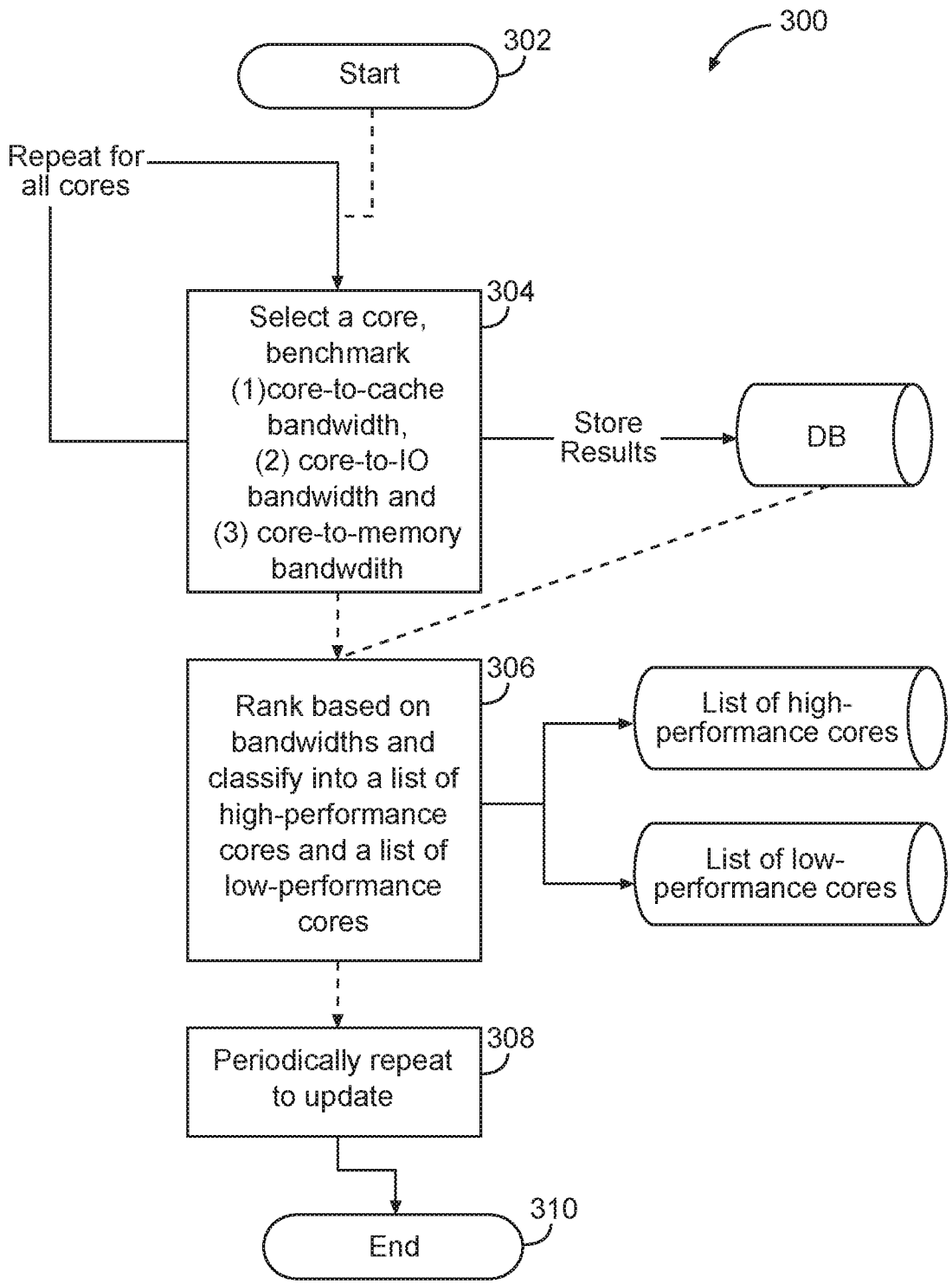
300
Start
302
Repeat for all cores
Select a core, benchmark (1)core-to-cache bandwidth, (2) core-to-IO bandwidth and (3) core-to-memory bandwdith
304
Store Results
DB
Rank based on bandwidths and classify into a list of high-performance cores and a list of low-performance cores
306
List of high-performance cores
List of low-performance cores
Periodically repeat to update
308
End
310

Fig. 3

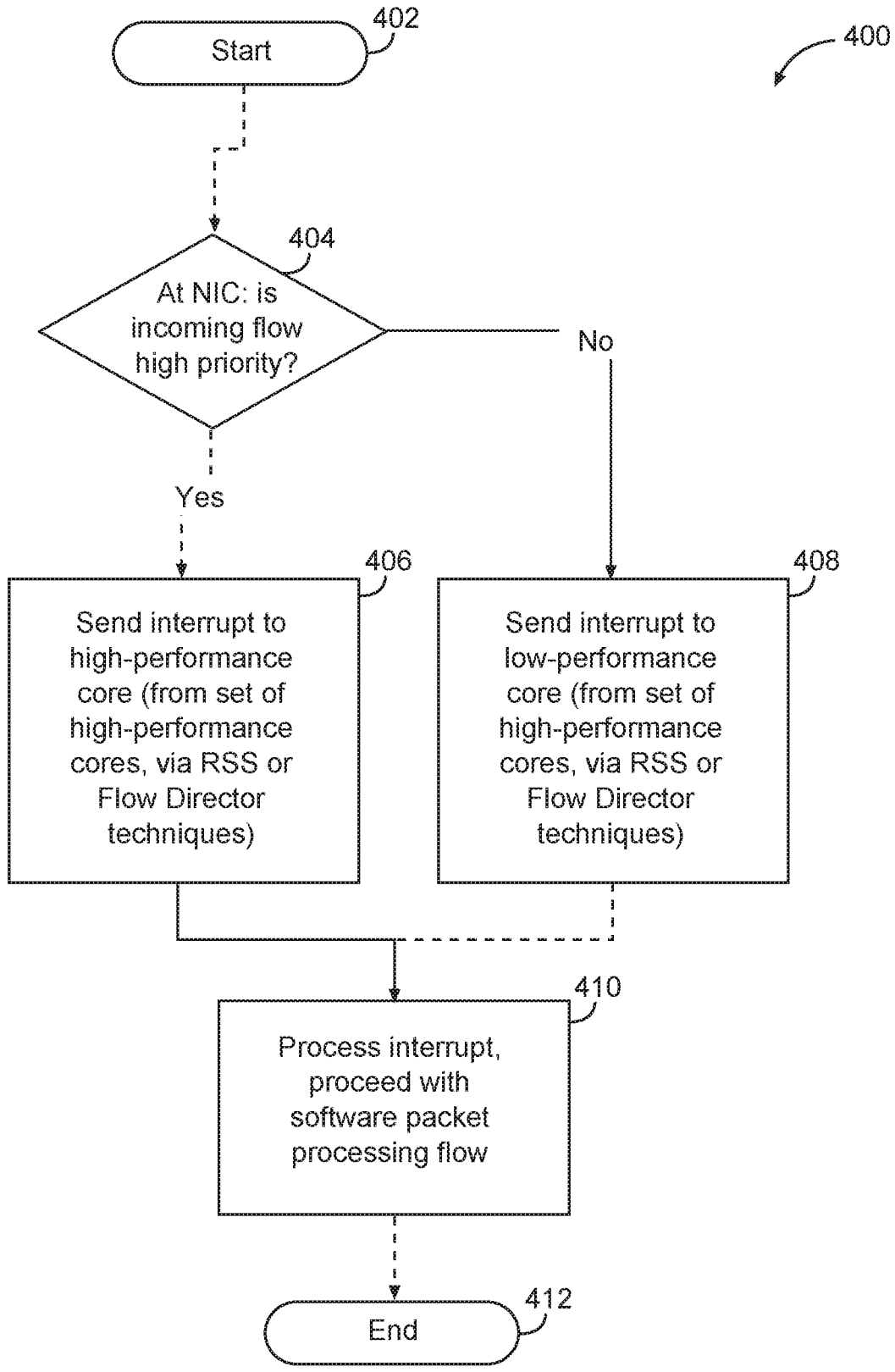
400
Start
402
At NIC: is incoming flow high priority?
404
No
Yes
Send interrupt to high-performance core (from set of high-performance cores, via RSS or Flow Director techniques)
406
Send interrupt to low-performance core (from set of high-performance cores, via RSS or Flow Director techniques)
408
Process interrupt, proceed with software packet processing flow
410
End
412

Fig. 4

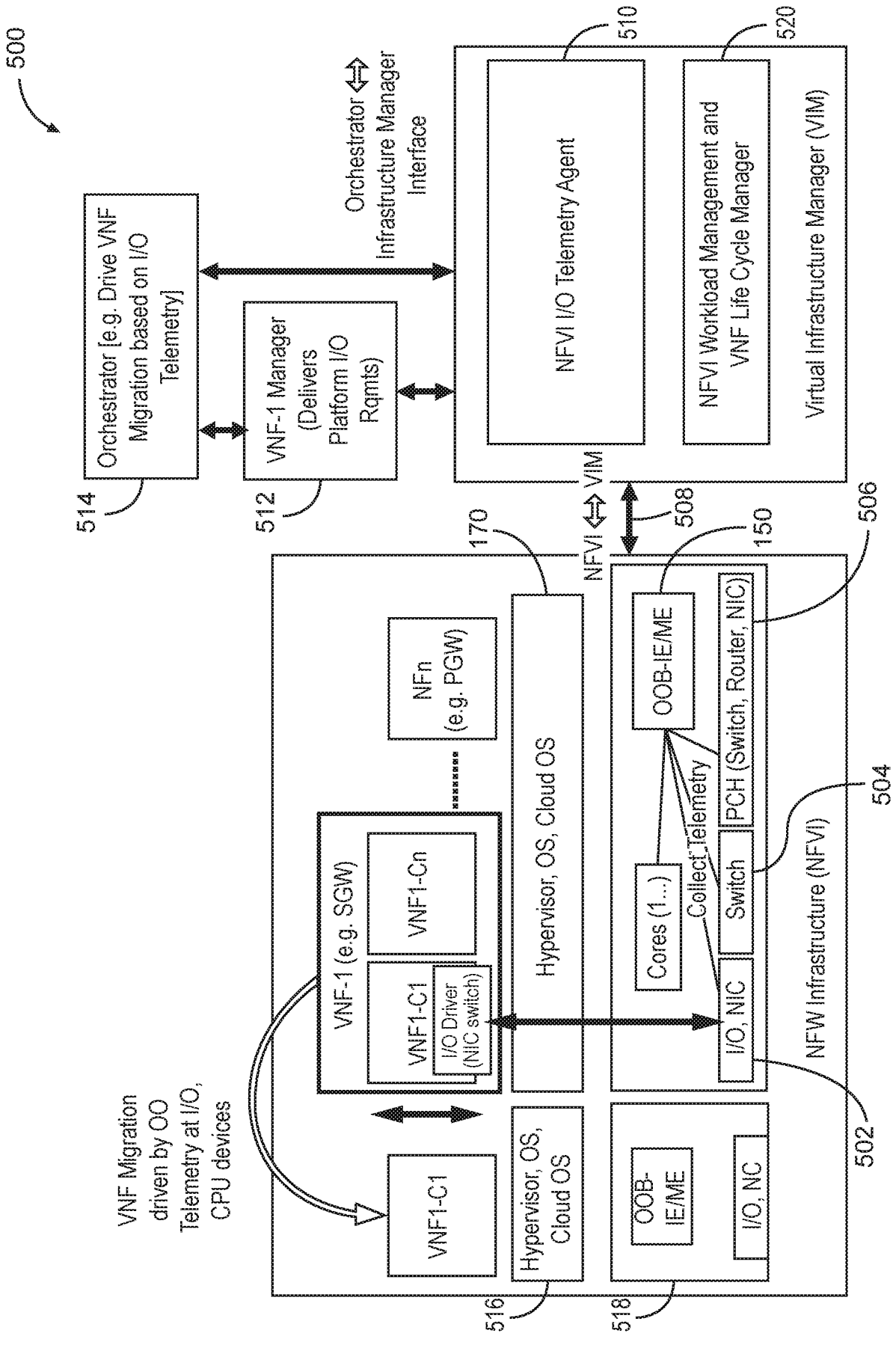
500
Orchestrator [e.g. Drive VNF Migration based on I/O Telemetry]
514
VNF-1 Manager (Delivers Platform I/O Rqmts)
512
Orchestrator ⇔ Infrastructure Manager Interface
NFVI I/O Telemetry Agent
510
NFVI Workload Management and VNF Life Cycle Manager
520
Virtual Infrastructure Manager (VIM)
NFVI ⇔ VIM
508
170
150
506
NFn (e.g. PGW)
VNF-1 (e.g. SGW)
VNF1-Cn
VNF1-C1
I/O Driver (NIC switch)
Hypervisor, OS, Cloud OS
OOB-IE/ME
Cores (1...)
Collect Telemetry
PCH (Switch, Router, NIC)
Switch
I/O, NIC
504
NFW Infrastructure (NFVI)
VNF Migration driven by OO Telemetry at I/O, CPU devices
VNF1-C1
Hypervisor, OS, Cloud OS
OOB-IE/ME
I/O, NC
502
516
518

Fig. 5

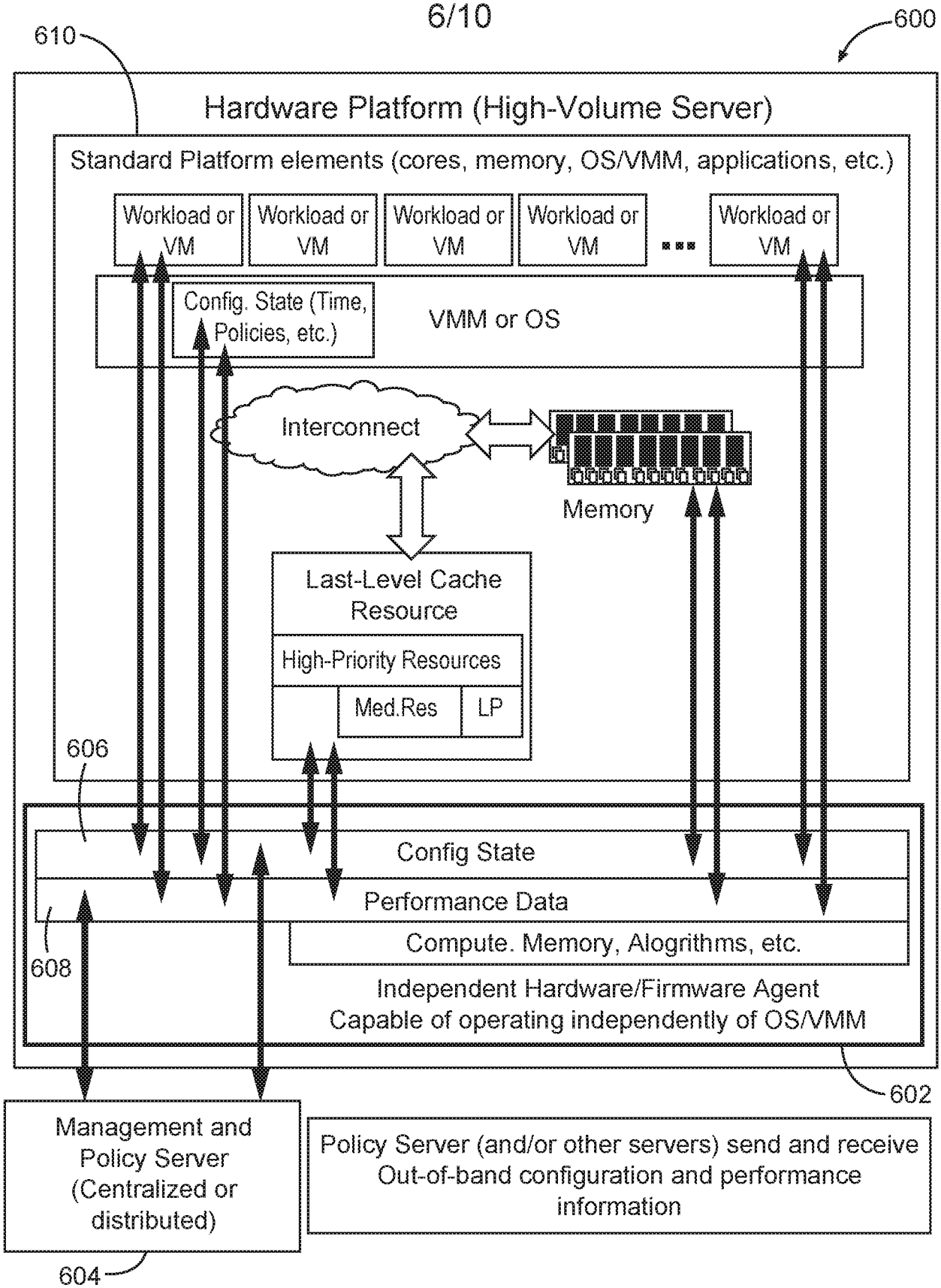
610
600
Hardware Platform (High-Volume Server)
Standard Platform elements (cores, memory, OS/VMM, applications, etc.)
Workload or VM
Workload or VM
Workload or VM
Workload or VM
Workload or VM
Config. State (Time, Policies, etc.)
VMM or OS
Interconnect
Memory
Last-Level Cache Resource
High-Priority Resources
Med.Res
LP
606
Config State
Performance Data
Compute. Memory, Alogrithms, etc.
608
Independent Hardware/Firmware Agent
Capable of operating independently of OS/VMM
602
Management and Policy Server (Centralized or distributed)
604
Policy Server (and/or other servers) send and receive Out-of-band configuration and performance information

Fig. 6

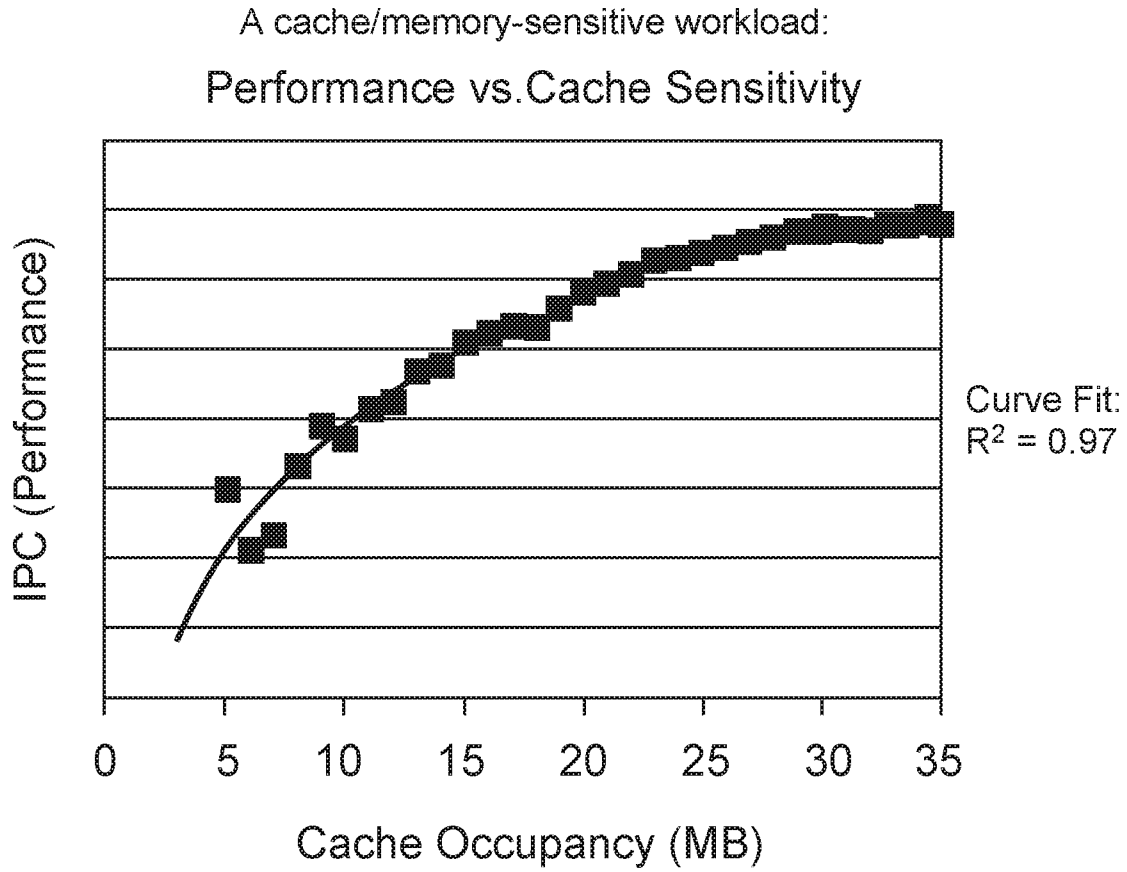
A cache/memory-sensitive workload:
Performance vs.Cache Sensitivity
IPC (Performance)
Curve Fit:
R2 = 0.97
0
5
10
15
20
25
30
35
Cache Occupancy (MB)

Fig. 7

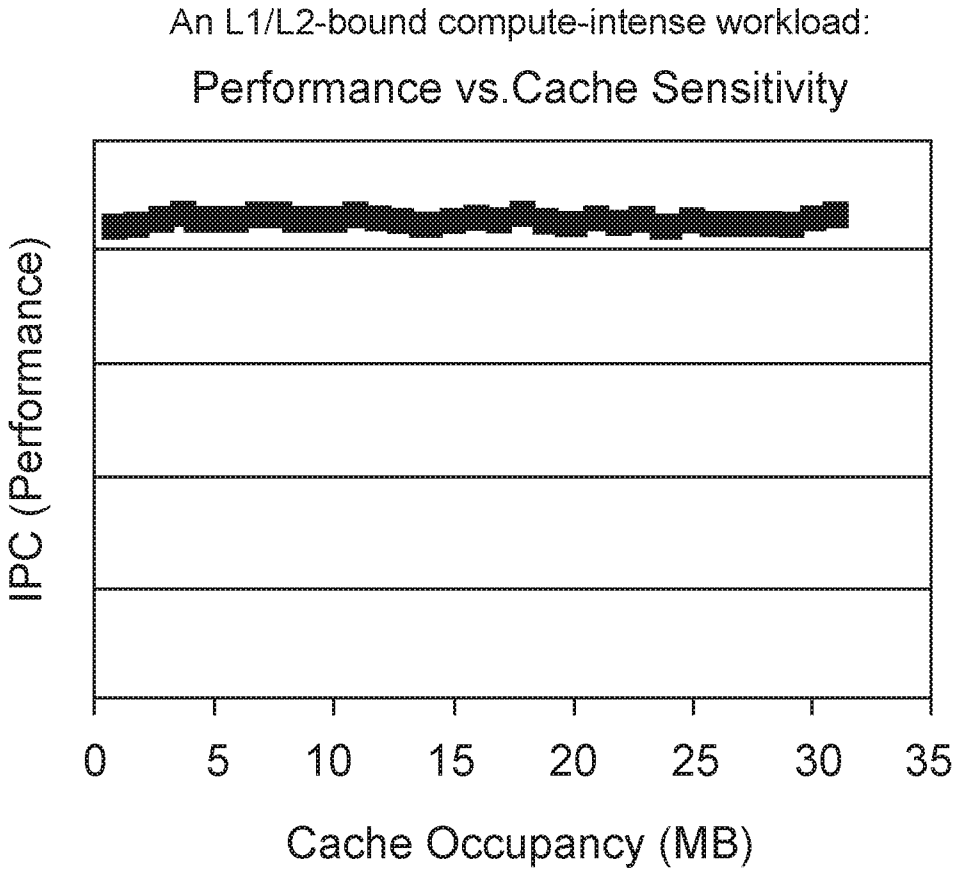
An L1/L2-bound compute-intense workload:
Performance vs.Cache Sensitivity
IPC (Performance)
0
5
10
15
20
25
30
35
Cache Occupancy (MB)

Fig. 8

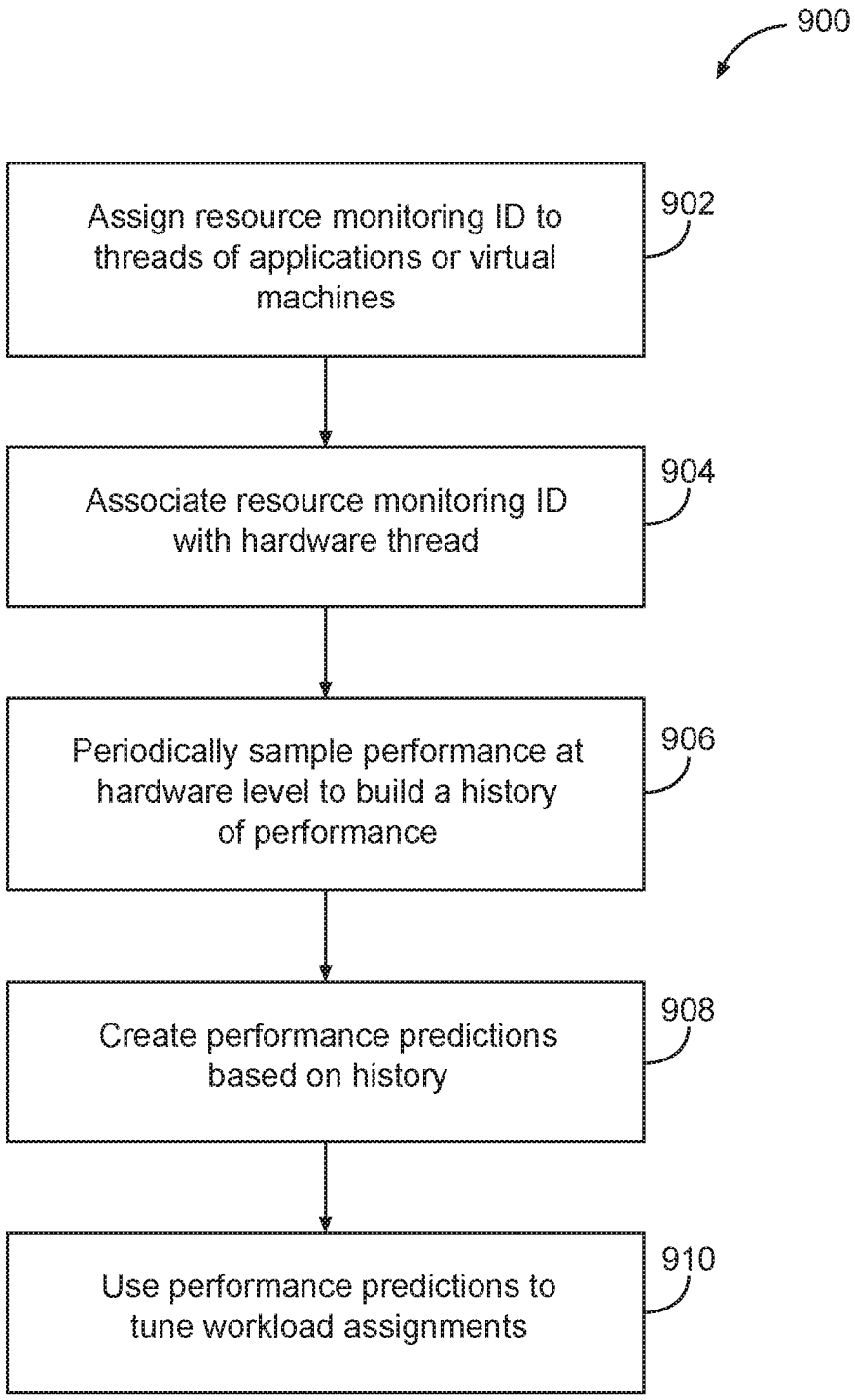
900
Assign resource monitoring ID to threads of applications or virtual machines
902
Associate resource monitoring ID with hardware thread
904
Periodically sample performance at hardware level to build a history of performance
906
Create performance predictions based on history
908
Use performance predictions to tune workload assignments
910

Fig. 9

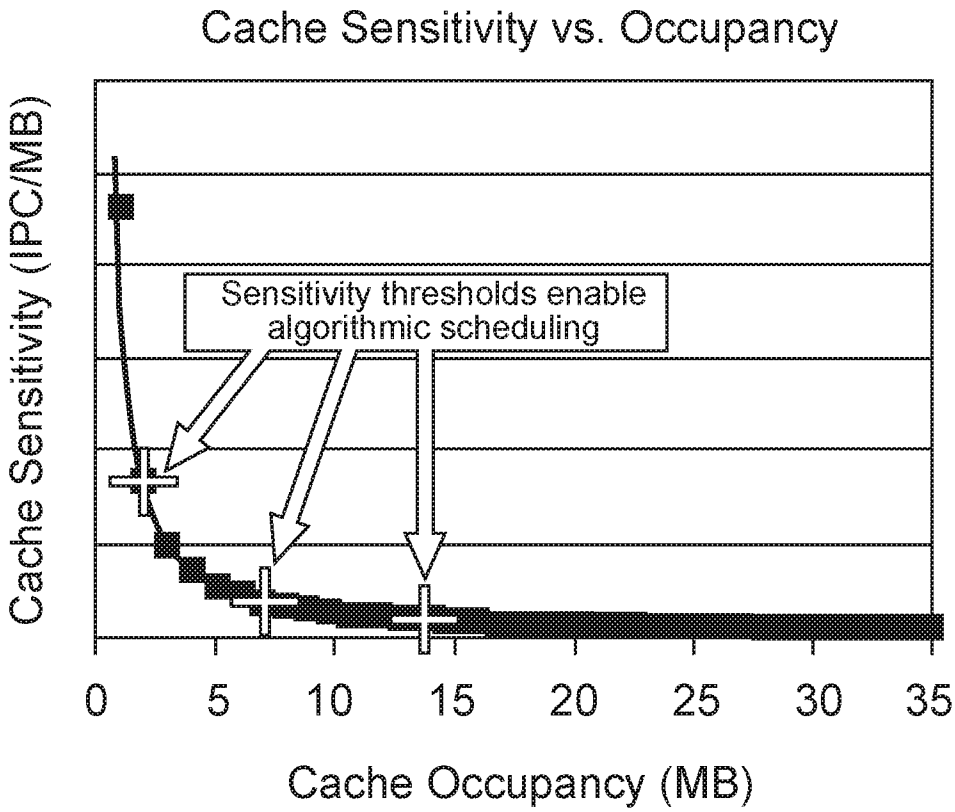
Cache Sensitivity vs. Occupancy
Cache Sensitivity (IPC/MB)
Sensitivity thresholds enable algorithmic scheduling
0
5
10
15
20
25
30
35
Cache Occupancy (MB)

Fig. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/048478

A. CLASSIFICATION OF SUBJECT MATTER

H04L 12/24(2006.01)i, H04L 12/931(2013.01)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 12/24; G06F 15/173; H04L 12/911; G06F 9/46; G06F 15/76; G06Q 30/08; G06F 12/08; H04L 12/931

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: out-of-band, telemetry, metrics, VM, workload, bandwidth, cache, performance, statistics, curve

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2009-0083737 A1 (SHMUEL BEN-YEHUDA et al.) 26 March 2009 See paragraphs [0005], [0018], [0020]-[0027], [0031]-[0034], [0041]-[0046], [0052]-[0053], [0068]; and figure 2.	1-4,9-10,16-17
Y		5-8,11-15,18-24
Y	US 2012-0151044 A1 (MICHAEL LUNA et al.) 14 June 2012 See paragraphs [0034], [0036], [0064], [0085], [0094], [0098], [0114], [0133], [0144]; and figure 1B.	5-8,11-15,18-24
A	US 2015-0235308 A1 (RACKSPACE US, INC.) 20 August 2015 See paragraphs [0031], [0053]-[0083]; and figures 1-2b.	1-24
A	US 2014-0122834 A1 (MRITTIKA GANGULI et al.) 01 May 2014 See paragraphs [0017]-[0033]; and figures 1-3.	1-24
A	US 2011-0066806 A1 (JATIN CHHUGANI et al.) 17 March 2011 See paragraphs [0022]-[0030] and figure 1.	1-24

☐ 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
28 November 2016 (28.11.2016)	**09 December 2016 (09.12.2016)**

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
KIM, Seong Woo
Telephone No. +82-42-481-3348

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/048478

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0083737 A1	26/03/2009	US 2012-124578 A1	17/05/2012
		US 8132185 B2	06/03/2012
		US 8413159 B2	02/04/2013
US 2012-0151044 A1	14/06/2012	CA 2798523 A1	31/05/2012
		CA 2798523 C	24/02/2015
		CA 2806527 A1	09/02/2012
		CA 2806529 A1	09/02/2012
		CA 2806529 C	09/12/2014
		CA 2806548 A1	09/02/2012
		CA 2806548 C	31/03/2015
		CA 2806549 A1	23/02/2012
		CA 2806549 C	28/10/2014
		CA 2806550 A1	09/02/2012
		CA 2806550 C	01/09/2015
		CA 2806557 A1	09/02/2012
		CA 2806557 C	07/10/2014
		CA 2857458 A1	09/02/2012
		CN 103404193 A	20/11/2013
		CN 103620576 A	05/03/2014
		EP 2596658 A1	29/05/2013
		EP 2599003 A1	05/06/2013
		EP 2599004 A1	05/06/2013
		EP 2599280 A2	05/06/2013
		EP 2599345 A2	05/06/2013
		EP 2599346 A2	05/06/2013
		EP 2599346 A4	27/11/2013
		EP 2599363 A2	05/06/2013
		EP 2635973 A2	11/09/2013
		EP 2636252 A2	11/09/2013
		EP 2636268 A2	11/09/2013
		EP 2661697 A2	13/11/2013
		GB 2495058 A	27/03/2013
		GB 2495066 A	27/03/2013
		GB 2495263 A	03/04/2013
		GB 2495263 B	16/04/2014
		GB 2495455 A	10/04/2013
		GB 2495463 A	10/04/2013
		GB 2495877 A	24/04/2013
		GB 2497012 A	29/05/2013
		GB 2499534 A	21/08/2013
		GB 2499741 A	28/08/2013
		GB 2499747 A	28/08/2013
		GB 2499936 A	04/09/2013
		GB 2500327 A	18/09/2013
		GB 2500333 A	18/09/2013
		GB 2500334 A	18/09/2013
		GB 2501416 A	23/10/2013
		GB 2504019 A	15/01/2014

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/048478

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		GB 2504634 A	05/02/2014
		GB 2506296 A	26/03/2014
		GB 2507005 A	16/04/2014
		GB 2507005 B	10/12/2014
		GB 2507426 A	30/04/2014
		GB 2510493 A	06/08/2014
		JP 2013-537754 A	03/10/2013
		JP 2013-539258 A	17/10/2013
		JP 2013-539259 A	17/10/2013
		JP 2013-541238 A	07/11/2013
		JP 5620578 B2	05/11/2014
		JP 5676762 B2	25/02/2015
		JP 5678185 B2	25/02/2015
		US 2012-0023236 A1	26/01/2012
		US 2012-0135726 A1	31/05/2012
		US 2012-0149352 A1	14/06/2012
		US 2012-0176968 A1	12/07/2012
		US 8838783 B2	16/09/2014
		US 8886176 B2	11/11/2014
		US 9077630 B2	07/07/2015
		US 9407713 B2	02/08/2016
		WO 2012-018430 A1	09/02/2012
		WO 2012-018431 A1	09/02/2012
		WO 2012-018477 A2	09/02/2012
		WO 2012-018477 A3	29/03/2012
		WO 2012-018479 A2	09/02/2012
		WO 2012-018479 A3	05/04/2012
		WO 2012-018556 A2	09/02/2012
		WO 2012-018556 A3	09/08/2012
		WO 2012-024030 A2	23/02/2012
		WO 2012-024030 A3	12/04/2012
		WO 2012-060995 A2	10/05/2012
		WO 2012-060995 A3	12/07/2012
		WO 2012-060996 A2	10/05/2012
		WO 2012-060996 A3	19/07/2012
		WO 2012-060997 A2	10/05/2012
		WO 2012-060997 A3	19/07/2012
		WO 2012-061430 A2	10/05/2012
		WO 2012-061430 A3	28/06/2012
		WO 2012-061433 A2	10/05/2012
		WO 2012-061433 A3	12/07/2012
		WO 2012-061437 A1	10/05/2012
		WO 2012-071283 A1	31/05/2012
		WO 2012-071384 A2	31/05/2012
		WO 2012-071384 A3	04/10/2012
		WO 2012-094675 A2	12/07/2012
		WO 2012-094675 A3	17/01/2013
		WO 2013-015835 A1	31/01/2013
US 2015-0235308 A1	20/08/2015	US 2013-304903 A1	14/11/2013

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/048478

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		WO 2014-116678 A1	31/07/2014
US 2014-0122834 A1	01/05/2014	None	
US 2011-0066806 A1	17/03/2011	US 8463820 B2	11/06/2013