

(43) International Publication Date
3 January 2014 (03.01.2014)

- (51) **International Patent Classification:**
G06F 9/46 (2006.01) *G06F 15/16* (2006.01)
- (21) **International Application Number:**
PCT/US2013/045990
- (22) **International Filing Date:**
14 June 2013 (14.06.2013)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
13/534,324 27 June 2012 (27.06.2012) US
- (71) **Applicant (for all designated States except US):** **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).
- (72) **Inventors; and**
- (71) **Applicants (for US only):** **JAIN, Niles K.** [IN/US]; 4377 NW Diamondback Dr., Beaverton, Oregon 97006 (US). **WILLKE, Theodore L.** [US/US]; 2105 North 26th Street, Tacoma, Washington 98403 (US). **DATTA, Kushal** [IN/US]; 2944 NW Moda Way, Hillsboro, Oregon 97124 (US). **YIGITBASI, Nezh** [TR/NL]; Parallelweg 10, NL-2613SL Delft (NL).
- (74) **Agents:** **KELLETT, Glen** et al.; Barnes And Thornburg, c/o CPA GLOBAL, P.O. Box 52050, Minneapolis, Minnesota 55402 (US).

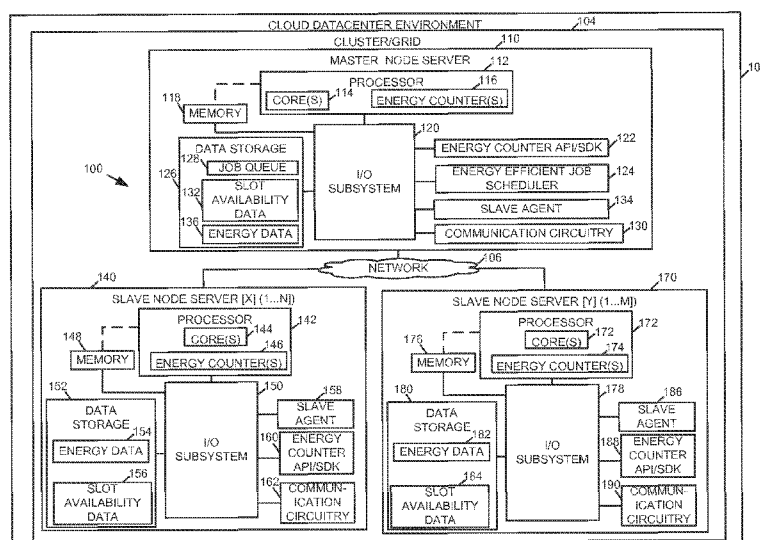
(81) **Designated States** (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

(54) **Title:** METHOD, SYSTEM, AND DEVICE FOR DYNAMIC ENERGY EFFICIENT JOB SCHEDULING IN A CLOUD COMPUTING ENVIRONMENT



(57) **Abstract:** A method, system, and device for energy efficient job scheduling in a datacenter computing environment includes a master node. The master node can periodically receive energy data from slave nodes and dynamically assign computing tasks to be executed by the slave nodes based on the energy data.

**METHOD, SYSTEM, AND DEVICE FOR DYNAMIC ENERGY
EFFICIENT JOB SCHEDULING IN A CLOUD COMPUTING ENVIRONMENT**

BACKGROUND

“Cloud” computing often refers to the provision of computing resources as a service, usually by a number of computer servers that are networked together at location(s) that are remote from the location from which the services are requested. A cloud datacenter refers to a physical arrangement of servers that make up a cloud (e.g., racks, rooms, etc.). In some cases, a particular portion of a datacenter may be implemented as a “cluster” or “grid.”

A cloud or cluster server or portions of its resources may be allocated physically or logically according to workload requirements. As an example, computing jobs that involve very large data sets and/or numerous computational tasks (such as big data analytics) may be distributed among multiple physical servers in a cluster and/or among multiple processes on the same server.

A scheduling system refers to computer software (e.g., middleware) used to allocate computing jobs among server resources in a cloud, cluster, or grid. For example, some scheduling systems designate one server as the “master” node of a cluster that includes a number of “slave” nodes, where the master node schedules tasks to be processed by its cluster in response to periodic “heartbeat” signals it receives from the slave nodes in its cluster.

BRIEF DESCRIPTION

The concepts described in the present disclosure are illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

FIG. 1 is a simplified block diagram of at least one embodiment of a dynamic energy efficient job scheduling system for managing server hardware resources in a cloud datacenter environment;

FIG. 2 is a simplified module diagram of at least one embodiment of executable modules of the system of FIG. 1; and

FIG. 3 is a simplified flow diagram of at least one embodiment of a method for dynamically scheduling jobs using energy efficiency metrics in the system of FIG. 1.

DETAILED DESCRIPTION

While the concepts of the present disclosure are susceptible to various
5 modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that there is no intent to limit the concepts of the present disclosure to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives consistent with the present disclosure and the
10 appended claims.

In the following description, numerous specific details such as logic implementations, opcodes, means to specify operands, resource partitioning/sharing/duplication implementations, types and interrelationships of system components, and logic partitioning/integration choices are set forth in order to provide a
15 more thorough understanding of the present disclosure. It will be appreciated by one skilled in the art, however, that embodiments of the disclosure may be practiced without such specific details. In other instances, control structures, gate level circuits and full software instruction sequences have not been shown in detail in order not to obscure the description of the of the concepts described herein. Those of ordinary skill in the art, with
20 the included descriptions, will be able to implement appropriate functionality without undue experimentation.

References in the specification to “one embodiment,” “an embodiment,” “an example embodiment,” etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily
25 include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

30 Embodiments of the concepts described herein may be implemented in hardware, firmware, software, or any combination thereof. Embodiments implemented in a computer system may include one or more point-to-point or bus-based interconnects between components. Embodiments of the concepts described herein may also be implemented as instructions carried by or stored on one or more machine-readable or

computer-readable storage media, which may be read and executed by one or more processors. A machine-readable or computer-readable storage medium may be embodied as any device, mechanism, or physical structure for storing or transmitting information in a form readable by a machine (e.g., a computing device). For example, a machine-readable
5 or computer-readable storage medium may be embodied as read only memory (ROM) device(s); random access memory (RAM) device(s); magnetic disk storage media; optical storage media; flash memory devices; mini- or micro-SD cards, memory sticks, and others.

In the drawings, specific arrangements or orderings of schematic elements, such as
10 those representing devices, modules, instruction blocks and data elements, may be shown for ease of description. However, it should be understood by those skilled in the art that the specific ordering or arrangement of the schematic elements in the drawings is not meant to imply that a particular order or sequence of processing, or separation of processes, is required. Further, the inclusion of a schematic element in a drawing is not
15 meant to imply that such element is required in all embodiments or that the features represented by such element may not be included in or combined with other elements in some embodiments.

In general, schematic elements used to represent instruction blocks may be implemented using any suitable form of machine-readable instruction, such as software or
20 firmware applications, programs, functions, modules, routines, processes, procedures, plug-ins, applets, widgets, code fragments and/or others, and that each such instruction may be implemented using any suitable programming language, library, application programming interface (API), and/or other software development tools. For example, some embodiments may be implemented using Java, C++, and/or other programming
25 languages. Similarly, schematic elements used to represent data or information may be implemented using any suitable electronic arrangement or structure, such as a register, data store, table, record, array, index, hash, map, tree, list, graph, file (of any file type), folder, directory, database, and/or others.

Further, in the drawings, where connecting elements, such as solid or dashed lines
30 or arrows, are used to illustrate a connection, relationship or association between or among two or more other schematic elements, the absence of any such connecting elements is not meant to imply that no connection, relationship or association can exist. In other words, some connections, relationships or associations between elements may not be shown in the drawings so as not to obscure the disclosure. In addition, for ease of illustration, a single

connecting element may be used to represent multiple connections, relationships or associations between elements. For example, where a connecting element represents a communication of signals, data or instructions, it should be understood by those skilled in the art that such element may represent one or multiple signal paths (e.g., a bus), as may
5 be needed, to effect the communication.

Referring now to FIG. 1, an illustrative system 100 for dynamic, energy efficient scheduling of computing jobs among a number of slave nodes 140, 170 of a cluster or grid 104 (referred to hereinafter as simply a “cluster” for ease of discussion) of a cloud datacenter environment 102 is managed by a master node 110. An energy efficient job
10 scheduler 124, embodied in the master node 110, communicates with slave agents 158, 186 of the slave nodes 140, 170 to determine which of the slave nodes 140, 170 may be the most energy efficient node to perform a given type of computing job or task. As explained in detail below, the energy efficient job scheduler 124 assigns computing jobs or tasks to the slave nodes 140, 170 based on information about the incoming job or task and
15 energy and availability information provided to the master node 110 periodically by the slave agents 158, 186 in real time. In this way, the energy efficiency of the cluster 104, and indeed the cloud datacenter environment 102, can be improved by matching computing jobs or tasks with the most appropriate energy efficient nodes for the particular type of job or task.

The illustrative cluster 104 is embodied as a collection of commodity computing devices that are connected to one another via the network 106 to function as a large multiprocessor. Further, the cluster 104 is embodied as a heterogeneous cluster. That is, the cluster 104 comprises a number (1...N) of slave node servers[X], where N is a positive integer and X represents one type of hardware configuration (e.g., processor, memory,
25 etc.), and the cluster 104 comprises a number (1...M) of slave node servers[Y], where M is a positive integer and Y represents another type of hardware configuration that is not the same as X. One example of a heterogeneous cluster is a datacenter that has multi-generation hardware.

As illustrated, the slave nodes 140, 170 are separate physical servers managed by
30 the master node server 110, but it should be appreciated by those skilled in the art that any or all of the “nodes” 110, 140, 170 may be logical rather than physical. That is, each physical server 110, 140, 170 may have one or more slave agents (e.g., 134, 158, 186) or a portion thereof running on it, using virtualization. In addition, the master node server 110 may host both the energy efficient job scheduler 124 and one or more slave agents 134,

using virtualization, in some embodiments. The servers 110, 140, 170 are in communication with one another via the network 106, as described in further detail below.

The cluster 104 is just one of many such groupings of computing devices that can make up the datacenter environment 102. In other words, the datacenter environment 102
5 may include a number of heterogeneous clusters such as the cluster 104, a number of homogeneous clusters (e.g., similar hardware), and/or other computing devices. In the illustrative embodiments, the job scheduling for each cluster 104 is managed by a master node such as the master node server 110. For illustration purposes, the cluster 104 is described herein as supporting distributed computing, and more particularly, distributed
10 computing on large data sets (sized in the petabytes range, in some examples) and/or highly distributable computations using a large number of server nodes (numbering in the thousands, in some examples). As such, the illustrative cluster 104 is described in the context of a software framework that is capable of processing highly distributable and/or data-intensive problems (e.g., big data analytics). For instance, in some embodiments, the
15 cluster 104 may be embodied using an implementation of a MapReduce framework, such as an APACHE HADOOP open-source implementation. However, it should be appreciated by those skilled in the art that the aspects of the present disclosure are in no way limited to HADOOP implementations, or even to MapReduce frameworks more generally. Rather, the present disclosure can be applied in any job scheduling context in
20 which computing jobs and/or tasks can be dynamically assigned to various hardware resources and where power and/or energy metrics are available.

In the illustrative embodiments, each of the node servers 110, 140, 170 is shown as having a similar or analogous hardware configuration. So as not to obscure the disclosure, the following description is presented in the context of the master node server 110 but
25 applies equally to the corresponding elements of each of the node servers 140, 170 having the same names, as shown in FIG. 1. The illustrative master node server 110 includes at least one processor 112, memory 118, an input/output (I/O) subsystem 120, an energy counter application programming interface (API) and/or software development toolkit (SDK) 122, the energy efficient job scheduler 124, at least one data storage device 126, a
30 job queue 128, communication circuitry 130, slot availability data 132, a slave agent 134, and energy data 136. The master node server 110 may be embodied as any suitable type of server computer or other computing device capable of performing the functions and features described herein as being performable by the master node server 110 or any of its components. For example, the master server 110 may be embodied as a computer, a

personal computer (PC), a desktop computer, a laptop computer, a notebook computer, a handheld computer, a server, a server array or server farm, a web server, a network server, an enterprise server, an Internet server, a work station, a mini-computer, a main frame computer, a supercomputer, a network appliance, a web appliance, a distributed computing
5 system, multiprocessor system, processor-based system, or combination of any of the foregoing.

The illustrative processor 112, 142, 172 includes at least one processor core 114, 144, 172 and one or more energy counters 116, 146, 174. The illustrative energy counters 116, 146, 174 monitor and output, in real time, information about the amount of power
10 and/or energy currently being used by the processor 112, 142, 172 (and/or the core 114, 144, 172), e.g., the number of watts used, and the processor's current frequency (e.g., in megahertz or gigahertz). While in the illustrative embodiments the energy counters 116, 146, 174 are available in the processor 112, 142, 172, it should be understood by those skilled in the art that, in some embodiments, the requisite energy and/or power data may
15 be obtained via special purpose tools or instrumentation available on the motherboard and/or by other devices that may be coupled to the processor 112, 142, 172.

In addition to an amount of cache memory, the processor 112, 142, 172 includes, or is otherwise communicatively coupled to, the memory 118, 148, 176. The memory 118, 148, 176 may be embodied as any type of suitable memory device, such as a dynamic
20 random access memory device (DRAM), synchronous dynamic random access memory device (SDRAM), double-data rate dynamic random access memory device (DDR SDRAM), and/or other volatile memory device.

The processor 112, 142, 172 is also communicatively coupled to the I/O subsystem 120, 150, 178. While not specifically shown, the illustrative I/O subsystem 120, 150, 178
25 may include a memory controller (e.g., a memory controller subsystem or northbridge), an input/output controller (e.g., an input/output controller subsystem or southbridge), and a firmware device. Of course, in other embodiments, I/O subsystems having other configurations may be used. For example, in some embodiments, the I/O subsystem 120, 150, 178 may form a portion of a system-on-a-chip (SoC) and be incorporated, along with
30 the processor 112, 142, 172 and other components of the servers 110, 140, 170, on a single integrated circuit chip. As such, it will be appreciated that each component of the I/O subsystem 120, 150, 178 may be located on a common integrated circuit chip in some embodiments.

The I/O subsystem 120 is communicatively coupled to the energy counter API/SDK 122, the energy efficient job scheduler 124, the data storage device 126, the communication circuitry 130, and the slave agent 134. The energy counter API/SDK 122, 160, 188 is embodied as one or more computerized programs, logic and/or instructions (e.g., software and/or firmware functions, libraries, drivers, operating system gadgets, and/or others) that enable software applications to import and export data generated by the energy counters 116, 146, 174. For example, some portions of the energy counter API/SDK 122, 160, 188 are configured to process the power and/or energy data generated by the energy counters 116, 146, 174, and convert it into energy metrics that can be used by software applications running on the master node server 110. Some examples of energy metrics include processor average power (Watts), cumulative energy consumed (joules), and instantaneous frequency. In the illustrative embodiments, the energy counter API/SDK 122, 160, 188 generates a CPU-related energy efficiency metric, performance per watt (perf/watt), where performance is defined as the reciprocal of task completion time (1/completion time), and generates an I/O-related energy efficiency metric, I/O operations per second (IOPS) per watt. The energy counter API/SDK 122, 160, 188 supplies these energy efficiency metrics to the corresponding slave agent 134, 158, 186 on a periodic basis, as described below. In addition, the energy efficiency metrics may be stored over time in the corresponding energy data 136, 154, 182.

The energy efficient job scheduler 124 is embodied as one or more computerized programs, logic and/or instructions (e.g., software and/or firmware) that are configured to process incoming computing jobs and/or tasks, store the incoming jobs/tasks in the job queue 128, and assign the jobs and/or tasks to the slave nodes 140, 170 (or a slave node running on the master node server 110), based on the energy efficiency data 136, 154, 182 and slot availability data 132, 156, 184 associated with the slave nodes 140, 170 and energy requirements associated with the particular job or task. Details relating to the operation of the energy efficient job scheduler 124 are described below with reference to FIGS. 2 and 3.

The slave agents 134, 158, 186 are each embodied as one or more computerized programs, logic and/or instructions (e.g., software and/or firmware) that are configured to periodically send “heartbeat” signals to the energy efficient job scheduler 124. Each heartbeat signal includes the current energy data 136, 154, 182 and slot availability data 132, 156, 184 associated with the server 110, 140, 170 issuing the heartbeat signals.

Portions of the data storage device 126, 152, 180 may be embodied as any suitable device for storing data and/or computer instructions, such as disk storage (e.g., hard disks), a network of physical and/or logical storage devices, and/or others. In the illustrative embodiment, the job queue 128, slot availability data 132, 156, 184 and energy data 136, 154, 182 reside in the corresponding data storage device 126, 152, 180. In addition, portions of the energy counter API/SDK 122, 160, 188, the energy efficient job scheduler 124, and/or the slave agent 134, 158, 186 may reside in the corresponding data storage device 126, 152, 180. Portions of the energy counter API/SDK 122, 160, 188, the energy efficient job scheduler 124, and/or the slave agent 134, 158, 186 may be copied to the corresponding memory 118, 148, 176 during operation, for faster processing or other reasons.

The communication circuitry 130, 162, 190 may be embodied as one or more devices and/or circuitry configured to enable communications between or among the master node server 110, the slave node server(s) 140, and/or the slave node server(s) 170, via the communication network 106. For example, the communication circuitry 130, 162, 190 may include one or more wired and/or wireless network interfaces or “cards” to facilitate communications over the wired and/or wireless portions of the network 106.

Although not specifically shown, the I/O subsystem 120, 150, 178 may be communicatively coupled to one or more peripheral device(s), such as a display, touchpad, keypad, microphone, speaker, and/or others, depending upon, for example, the intended use of the respective server 110, 140, 170. Further, it should be appreciated that the master node server 110, and/or any of the slave node servers 140, 170, may include other components, sub-components, and devices not illustrated in FIG. 1 for clarity of the description.

The network 106 may be embodied as any type of wired and/or wireless telecommunications network. For example, the network 106 may be embodied as or otherwise include one or more public or private cellular networks, telephone, Digital Subscriber Line (DSL) or cable networks, local or wide area networks, publicly available global networks (e.g., the Internet), or any combination thereof. For example, in some embodiments, the network 106 may be embodied as or otherwise include a Global System for Mobile Communications (GSM) cellular network. Additionally, the network 106 may include any number of additional devices as needed to facilitate communication between or among the master node server 110 and/or the slave node servers 140, 170, such as routers, switches, intervening computers and/or others. Any suitable communication

protocol (e.g., TCP/IP) may be used to effect communication over the network 106, depending on, for example, the particular type or configuration of the network 106.

In general, the components of the master node server 110 and the slave node servers 140, 170 are communicatively coupled as shown in FIG. 1, by one or more signal paths. Such signal paths may be embodied as any type of wired or wireless signal paths capable of facilitating communication between the respective devices. For example, the signal paths may be embodied as any number of wires, printed circuit board traces, via, bus, point-to-point interconnects, intervening devices, and/or the like. Also, generally speaking, some of the components of the computing devices described above may be incorporated on a motherboard while other components may be communicatively coupled to the motherboard via, for example, a peripheral port.

Referring now to FIG. 2, in an embodiment 200 of the system 100, each of the illustrative energy efficient job scheduler 124 and the slave agents 134, 158, 186 is embodied as one or more computerized modules (e.g., software and/or firmware). The energy efficient job scheduler 124 periodically receives computing jobs or tasks, e.g., from client applications 210 running on end user computing devices and/or other computing devices connected to the network 106. One example of a job or task that may be received from a client application 210 is a web search that is initiated by an end user at a computing device that is connected to the network 106 via the Internet. Some other examples of jobs that may be handled by the energy efficient job scheduler 124 include web crawling, text tokenizers, page ranking, document processing, and clickstream analysis.

In the illustrative embodiment, a job(z), where z is a positive integer, can be broken down by the energy efficient job scheduler 124 into a number of tasks(1...N), where N is a positive integer. Each of the tasks (1...N) can be classified as a certain type of task (e.g., T1, T2). In addition, the tasks (1...N) that make up the job(z) can be distributed among multiple slave nodes 140, 170 for execution. In the web search illustration, the job of searching the web for a specific combination of search terms input by a user may be broken down into smaller tasks, where each task involves searching a particular domain of the Internet or conducting a search on a specific permutation of the search term, for example.

In the context of the illustrative MapReduce framework, input data associated with a job (e.g., data that may be used by the job and/or one or more tasks, which may reside in a file system, such as the file system known as the HADOOP File System or HDFS), is split and distributed among the cluster nodes. The job is divided into a number of tasks,

based on the size of the input data. Each task can be classified as either a “map” task or a “reduce” task. Generally speaking, map tasks are smaller, sub-tasks that can be distributed to the slave nodes and performed thereby, while reduce tasks are tasks that collect and combine the results of all of the map tasks to form the output resulting from completion of the job. As such, map tasks are typically computation-intensive while reduce tasks tend to be more I/O intensive. The number of map tasks and the number of input splits may have a 1:1 correspondence.

More generally, that is, irrespective of the framework on which the energy efficient job scheduler 124 may be based, any job may have tasks or sub-portions that can be classified similarly, as computation-intensive, I/O intensive, or using any other suitable classification scheme. In other words, the number of task types may be determined, and tasks may be classified in any number of ways, according to the requirements of a particular design and/or implementation of the system 100.

As noted above, the slave agents [X](N) 158 and [Y](M) 186 send periodic heartbeat signals to the energy efficient job scheduler 124. In the illustrative embodiments, each heartbeat signal includes a data structure containing the current energy efficiency data 136, 154, 182 for the server issuing the heartbeat signal, as well as the server’s slot availability data 132, 156, 184. In some MapReduce implementations, the slot availability data 132, 156, 184 includes information relating to the number of “slots” that are available at the server issuing the heartbeat signal to receive map tasks and the number of slots that are available to receive reduce tasks. More generally, in other embodiments, the slot availability data 132, 156, 184 simply includes data that gives an indication of the corresponding server’s capacity to accept new jobs or tasks. In response to a heartbeat signal, the energy efficient job scheduler 124 traverses the job queue 128 in priority order and determines which tasks to assign to the server from which the current heartbeat signal was received.

Referring now to FIG. 3, an illustrative method 300 executable as computerized logic and/or instructions by the various modules of the energy efficient job scheduler 124 is shown. At block 310, the method 300 receives a heartbeat signal from one of the slave agents 134, 158, 186. As noted above, the heartbeat signal includes the energy efficiency metrics (e.g., records per joule and IOPS/watt) and slot availability data 132, 156, 184 associated with the server 110, 140, 170 (“slave node(i)”) on which the slave agent 134, 158, 186 is running.

Different types of computing tasks may have different energy efficiency characteristics, as mentioned above. At block 312, the method 300 determines a given type of computing task, T, to use as a basis for evaluating the energy efficiency of the slave node(i). In the illustrative method 300, two task types, T1 or T2 (e.g., map or
5 reduce) are shown. However, in other embodiments, incoming tasks may be classified as one of any number of task types. For each heartbeat signal, the method 300 processes tasks of one of the tasks types (e.g., T1 or map tasks) first, and then processes tasks of the other task type(s), in the illustrative embodiments. In some embodiments, the processing of at least some tasks by the method 300 may be prioritized in other ways, not prioritized
10 at all, or performed concurrently rather than sequentially. Block 312 deals with a particular task type T (e.g., T1 or T2) depending on the task type that is selected for processing.

At block 314, the method 300 determines whether the current slave node, “node(i)” (that is, the node from which the method 300 received a heartbeat signal at block 310) is
15 the most energy efficient node for the task type determined at block 312. If the node(i) is not the most energy efficient node for the task type or does not have any slots available for that task type, then the method 300 returns to block 310 without scheduling any tasks to the node(i). As an example, if the task type T relates to map tasks (or, tasks that are computation-intensive), and the node(i) has a good (e.g., high) value for the energy
20 efficiency metric for computation-intensive tasks (e.g., a high number of records per joule), then the node(i) may be considered the most energy efficient node for the task type. Conversely, if the task type T is map, but the node(i) has a high value for the energy efficiency metric for I/O intensive tasks (e.g., a high number of IOPS/watt) but a lower value for the energy efficiency metric for computation-intensive tasks, then the node(i)
25 may not be considered the most energy efficient node for the task type. In determining whether a node(i) is the most energy efficient node for a task type T, the method 300 uses a greedy heuristic algorithm, e.g., a strategy of looking for the best or optimal choice on a local level (e.g., within the cluster 104) rather than on a global level (e.g., across clusters or within the entire datacenter environment 102). In addition, at block 314, the method
30 300 determines, from the slot availability data 132, 156, 184, whether the node(i) has the capacity (e.g., available slots) to accept tasks of the task type T.

If the node(i) is the most energy efficient node for the task type T, and the node(i) has the capacity to accept tasks of the task type T, then the method 300 determines a number of tasks to schedule to the node(i) based on the slot availability data 132, 156, 184

for the node(i), and proceeds to schedule tasks of type T to that node until the node(i) is no longer the most energy efficient node (e.g., until the value of the node(i)'s energy efficiency metric is no longer considered the "best" value for the task type T), or the node(i) no longer has the capacity to accept tasks of task type T. If the node(i) is not the most energy efficient node for the task type T, or does not have available slots for the task type T, the method 300 returns to block 310.

At block 316, the job queue 128, which contains data relating to the runnable jobs that have been received by the energy efficient job scheduler 124 and the number of running tasks associated with each job, is sorted to give higher priority to those jobs that have a higher number of running tasks (e.g., for fairness). At block 318, a job j and data relating to its associated tasks is read from the top of the job queue 128. At block 320, the method 300 determines whether the job j has any unlaunched tasks t (e.g., tasks that are runnable but not yet running) that need to be scheduled. If the job j does not have any unlaunched tasks t, the method 300 returns to block 318 and begins processing of the next job j in the job queue 128. If the job j does have unlaunched tasks t, the method processes tasks t that are of task type T1 first, and then processes tasks t that are of task type T2. So, if the method 300 is currently processing tasks of task type T1 (determined at block 322), the method 300 proceeds to block 324. If the method is not currently processing tasks of task type T1 (determined at block 322), the method proceeds to block 330, where it is determined whether tasks of type T2 are currently being processed.

If tasks of task type T1 are currently being processed, then at block 324, the method 300 determines whether any of the unlaunched tasks t of the current job j are of the task type T1, and if so, evaluates the data locality of each of the unlaunched tasks t that are of the task type T1. The data locality analysis can be used to assign tasks to nodes on which the associated input data (e.g., "split") is already located, to avoid unnecessary data transfer I/O operations, or for other reasons. In some embodiments, the data locality analysis may consider not only node-level locality, but also rack-level and off-rack locality. For instance, considering rack-level locality, even if the input data for a task is not located on the node(i) but is located on another node within the same rack as the node(i), the method 300 may schedule the task to the node(i) in some embodiments. In other embodiments, the data locality analysis may require tasks to be assigned to the node that contains the associated input split (e.g., ignoring rack- and off-rack locality). As should be appreciated by those skilled in the art, the degree to which data locality is

considered by the method 300 can be varied according to the requirements of a particular design or implementation of the method 300.

Referring to blocks 326 and 328, the method first assigns the tasks *t* of task type T1 that meet the data locality criteria as determined at block 324, for execution by the node(*i*), at block 326, and then, at block 328, assigns any other tasks *t* of task type T1 to the node(*i*), according to the number of slots remaining available at the node(*i*). Once all of the tasks *t* of task type T1 have been processed, at block 332, the method 300 schedules tasks *t* of task type T2 to the node(*i*) in accordance with the energy efficiency of the node(*i*) and available slots for the task type T2. While described herein as being performed in a more or less sequential manner, it should be understood that the scheduling of tasks of types T1 and T2 may be done concurrently, in some embodiments.

EXAMPLES

Illustrative examples of the devices, systems, and methods disclosed herein are provided below. An embodiment of the devices, systems, and methods may include any one or more, and any combination of, the examples described below.

In an example, a master node for use in a datacenter computing environment comprising a communication network and a plurality of slave nodes includes an energy efficient job scheduler to periodically receive energy data from the slave nodes, periodically receive computing jobs including one or more unlaunched computing tasks, determine a task type for an unlaunched computing task, and assign the unlaunched computing task to a slave node in response to the energy data received from the slave node based on the task type.

In an example, the energy efficient job scheduler may be embodied as middleware of the master node and the master node may be embodied as a server computer. In an example, the energy data received from a slave node by the energy efficient job scheduler may indicate whether the slave node is more energy efficient for processor-intensive tasks or for input-output intensive tasks. In an example, the energy efficient job scheduler may assign the unlaunched computing task to a first slave node that is more energy efficient for processor-intensive tasks in response to determining that the task type is a first task type and may assign the unlaunched computing task to a second slave node that is more energy efficient for input-output intensive tasks in response to determining that the task type is a second task type different than the first task type. In an example, the energy efficient job scheduler may implement a MapReduce framework, where the first task type is “Map” and the second task type is “Reduce.” In an example, the energy efficient job scheduler

may determine whether data associated with the unlaunched computing task is local to a slave node or a rack containing the slave node, and may assign the unlaunched computing task to the slave node in response to determining that data associated with the unlaunched computing task is local to the slave node or the rack containing the slave node. In an example, the energy efficient job scheduler may maintains a job queue including computing jobs having unlaunched computing tasks, and may periodically receive slot availability data from the slave nodes, where the slot availability data may indicate a number of slots a slave node has available for unlaunched computing tasks, and may determine a number of unlaunched computing tasks to assign to the slave node in response to the number of available slots. In an example, the energy efficient job scheduler may periodically receive, from the slave nodes, first slot availability data indicating a first number of slots a slave node has available for unlaunched tasks of a first task type and second slot availability data indicating a second number of slots the slave node has available for unlaunched tasks of a second task type, and may assign the unlaunched computing tasks to the slave nodes based on the task type and the first and second slot availability data. In an example, the plurality of slave nodes may include a heterogeneous cluster of computing devices, and the energy efficient job scheduler may selectively assign the unlaunched computing tasks to computing devices in the heterogeneous cluster.

In another example, a slave node for use in a datacenter computing environment including a communication network and a master node configured to assign computing tasks to a plurality of slave nodes includes a slave agent to periodically send energy data to the master node and receive unlaunched computing tasks from the master node for execution by the slave node in response to the energy data sent by the slave node to the master node, where the received unlaunched computing tasks each have a task type that corresponds to the energy data sent by the slave node to the master node. In an example, the master node and the slave node may be embodied as server computers. In an example, the master node and the slave node may be embodied as virtual nodes running on the same server computer. In an example, the energy data may include a first energy metric including records per joule and a second energy metric including input-output operations per second per watt. In an example, the slave agent may be configured to periodically send slot availability data for each task type to the master node and receive unlaunched computing tasks from the master node for execution by the slave node in response to the slot availability data.

In another example, a system for energy efficient job scheduling in a datacenter computing environment includes a plurality of slave nodes each comprising a slave agent; and a master node comprising an energy efficient job scheduler to periodically receive energy data from the slave agents and assign unlaunched computing tasks each having a task type to the slave nodes in response to the energy data received from the slave agents and based on the task type.

In an example, the plurality of slave nodes may be embodied as a heterogeneous cluster of computing devices comprising at least one first computing device that is more energy efficient for processor-intensive computing tasks and at least one second computing device that is more energy efficient for input-output intensive computing tasks. In an example, the master node may assign unlaunched computing tasks of a first task type to slave nodes embodied as at least one computing device that is more energy efficient for processor-intensive computing tasks. In an example, the master node may receive locality data associated with the unlaunched tasks from the slave agents and may assign unlaunched computing tasks of the first type to slave nodes based on the locality data. In an example, the master node may assign unlaunched computing tasks of a second task type to slave nodes embodied as at least one computing device that is more energy efficient for input-output intensive computing tasks. In an example, the master node may execute a greedy heuristic algorithm to assign the unlaunched computing tasks to the slave nodes.

In another example, a method for scheduling a plurality of unlaunched computing tasks to be executed by one or more slave nodes in a datacenter computing environment includes periodically receiving, at a master node of the datacenter computing environment, energy and availability data from each of the slave nodes, and, in response to receiving energy and availability data from one of the slave nodes, determining whether the slave node is an energy efficient node for a first type of computing task.

In an example, the method may include, in response to determining that the slave node is an energy efficient node for the first type of computing task, determining a number of unlaunched computing tasks of the first type to assign to the slave node based on the availability data for the slave node. In an example, the method may include assigning the determined number of unlaunched tasks of the first type to the slave node according to locality data associated with each of the determined number of unlaunched tasks of the first type. In an example, the method may include, in response to determining that the slave node is not an energy efficient node for the first type of computing task, determining whether the slave node is an energy efficient node for a second type of computing task. In

an example, the method may include, in response to determining that the slave node is an energy efficient node for the second type of computing task, determining a number of unlaunched tasks of the second type to assign to the slave node based on the availability data for the slave node. In an example, the method may include assigning the determined
5 number of unlaunched tasks of the second type to the slave node. In an example, the method may include repeating the determining whether the slave node is an energy efficient node for a first type of computing task and assigning the determined number of unlaunched tasks of the first type to the slave node until it is determined that the slave node is no longer an energy efficient node for the first type of computing task.

10 In another example, at least one machine accessible storage medium includes a plurality of instructions that in response to being executed result in a computing device, periodically receiving, at a master node of the datacenter computing environment, energy and availability data from each of the slave nodes, and, in response to receiving energy and availability data from one of the slave nodes, determining whether the slave node is an
15 energy efficient node for a first type of computing task.

In an example, the at least one computer accessible storage medium may include, in response to determining that the slave node is an energy efficient node for the first type of computing task, determining a number of unlaunched computing tasks of the first type to assign to the slave node based on the availability data for the slave node. In an
20 example, the at least one computer accessible storage medium may include assigning the determined number of unlaunched tasks of the first type to the slave node according to locality data associated with each of the determined number of unlaunched tasks of the first type. In an example, the at least one computer accessible storage medium may include, in response to determining that the slave node is not an energy efficient node for
25 the first type of computing task, determining whether the slave node is an energy efficient node for a second type of computing task. In an example, the at least one computer accessible storage medium may include, in response to determining that the slave node is an energy efficient node for the second type of computing task, determining a number of unlaunched tasks of the second type to assign to the slave node based on the availability
30 data for the slave node; and assigning the determined number of unlaunched tasks of the second type to the slave node. In an example, the at least one computer accessible storage medium may include repeating the determining whether the slave node is an energy efficient node for a first type of computing task and assigning the determined number of

unlaunched tasks of the first type to the slave node until it is determined that the slave node is no longer an energy efficient node for the first type of computing task.

While the concepts of the present disclosure have been illustrated and described in detail in the drawings and foregoing description, such an illustration and description is to
5 be considered as exemplary and not restrictive in character, it being understood that only illustrative embodiments have been shown and described and that all changes and modifications consistent with the disclosure and recited claims are desired to be protected.

CLAIMS

1. A master node for use in a datacenter computing environment comprising a communication network and a plurality of slave nodes, the master node comprising:

an energy efficient job scheduler to periodically receive energy data from the slave
5 nodes, periodically receive computing jobs comprising one or more unlaunched computing tasks, determine a task type for an unlaunched computing task, and assign the unlaunched computing task to a slave node in response to the energy data received from the slave node based on the task type.

2. The master node of claim 1, wherein the energy efficient job scheduler is
10 embodied as middleware of the master node and the master node is embodied as a server computer.

3. The master node of claim 1, wherein the energy data received from a slave node by the energy efficient job scheduler indicates whether the slave node is more energy efficient for processor-intensive tasks or for input-output intensive tasks.

15 4. The master node of claim 3, wherein the energy efficient job scheduler assigns the unlaunched computing task to a first slave node that is more energy efficient for processor-intensive tasks in response to determining that the task type is a first task type and assigns the unlaunched computing task to a second slave node that is more energy efficient for input-output intensive tasks in response to determining that the task
20 type is a second task type different than the first task type.

5. The master node of claim 4, wherein the energy efficient job scheduler implements a MapReduce framework, the first task type is "Map" and the second task type is "Reduce."

6. The master node of claim 1, wherein the energy efficient job scheduler
25 determines whether data associated with the unlaunched computing task is local to a slave node or a rack containing the slave node, and assigns the unlaunched computing task to the slave node in response to determining that data associated with the unlaunched computing task is local to the slave node or the rack containing the slave node.

7. The master node of claim 1, wherein the energy efficient job scheduler
30 maintains a job queue comprising a plurality of computing jobs having unlaunched computing tasks, periodically receives slot availability data from the slave nodes, the slot availability data indicating a number of slots a slave node has available for unlaunched computing tasks, and determines a number of unlaunched computing tasks to assign to the slave node in response to the number of available slots.

8. The master node of claim 7, wherein the energy efficient job scheduler periodically receives, from the slave nodes, first slot availability data indicating a first number of slots a slave node has available for unlaunched tasks of a first task type and second slot availability data indicating a second number of slots the slave node has available for unlaunched tasks of a second task type, and assigns the unlaunched computing tasks to the slave nodes based on the task type and the first and second slot availability data.

9. The master node of claim 1, wherein the plurality of slave nodes comprises a heterogeneous cluster of computing devices, and the energy efficient job scheduler selectively assigns the unlaunched computing tasks to computing devices in the heterogeneous cluster.

10. A slave node for use in a datacenter computing environment comprising a communication network and a master node configured to assign computing tasks to a plurality of slave nodes, the slave node comprising:

15 a slave agent to periodically send energy data to the master node and receive unlaunched computing tasks from the master node for execution by the slave node in response to the energy data sent by the slave node to the master node, wherein the received unlaunched computing tasks each have a task type that corresponds to the energy data sent by the slave node to the master node.

20 11. The slave node of claim 10, wherein the master node and the slave node are embodied as server computers.

12. The slave node of claim 10, wherein the master node and the slave node are embodied as virtual nodes running on the same server computer.

25 13. The slave node of claim 10, wherein the energy data comprises a first energy metric including records per joule and a second energy metric including input-output operations per second per watt.

14. The slave node of claim 10, wherein the slave agent is configured to periodically send slot availability data for each task type to the master node and receive unlaunched computing tasks from the master node for execution by the slave node in response to the slot availability data.

30 15. A method for scheduling a plurality of unlaunched computing tasks to be executed by one or more slave nodes in a datacenter computing environment, the method comprising:

periodically receiving, at a master node of the datacenter computing environment, energy and availability data from each of the slave nodes, and, in response to receiving energy and availability data from one of the slave nodes, determining whether the slave node is an energy efficient node for a first type of computing task.

5 16. The method of claim 15, comprising, in response to determining that the slave node is an energy efficient node for the first type of computing task, determining a number of unlaunched computing tasks of the first type to assign to the slave node based on the availability data for the slave node.

10 17. The method of claim 16, comprising assigning the determined number of unlaunched tasks of the first type to the slave node according to locality data associated with each of the determined number of unlaunched tasks of the first type.

 18. The method of claim 17, comprising, in response to determining that the slave node is not an energy efficient node for the first type of computing task, determining whether the slave node is an energy efficient node for a second type of computing task.

15 19. The method of claim 18, comprising, in response to determining that the slave node is an energy efficient node for the second type of computing task, determining a number of unlaunched tasks of the second type to assign to the slave node based on the availability data for the slave node; and assigning the determined number of unlaunched tasks of the second type to the slave node.

20 20. The method of claim 17, comprising repeating the determining whether the slave node is an energy efficient node for a first type of computing task and assigning the determined number of unlaunched tasks of the first type to the slave node until it is determined that the slave node is no longer an energy efficient node for the first type of computing task.

25 21. At least one machine accessible storage medium comprising a plurality of instructions that in response to being executed result in a computing device:

 periodically receiving, at a master node of a datacenter computing environment, energy and availability data from each of a plurality of slave nodes, and, in response to receiving energy and availability data from one of the slave nodes, determining whether
30 the slave node is an energy efficient node for a first type of computing task.

 22. The at least one computer accessible storage medium of claim 21, comprising, in response to determining that the slave node is an energy efficient node for the first type of computing task, determining a number of unlaunched computing tasks of the first type to assign to the slave node based on the availability data for the slave node.

23. The at least one computer accessible storage medium of claim 22, comprising assigning the determined number of unlaunched tasks of the first type to the slave node according to locality data associated with each of the determined number of unlaunched tasks of the first type.

5 24. The at least one computer accessible storage medium of claim 23, comprising, in response to determining that the slave node is not an energy efficient node for the first type of computing task, determining whether the slave node is an energy efficient node for a second type of computing task.

10 25. The at least one computer accessible storage medium of claim 24, comprising, in response to determining that the slave node is an energy efficient node for the second type of computing task, determining a number of unlaunched tasks of the second type to assign to the slave node based on the availability data for the slave node; and assigning the determined number of unlaunched tasks of the second type to the slave node.

15 26. The at least one computer accessible storage medium of claim 23, comprising repeating the determining whether the slave node is an energy efficient node for a first type of computing task and assigning the determined number of unlaunched tasks of the first type to the slave node until it is determined that the slave node is no longer an energy efficient node for the first type of computing task.

20

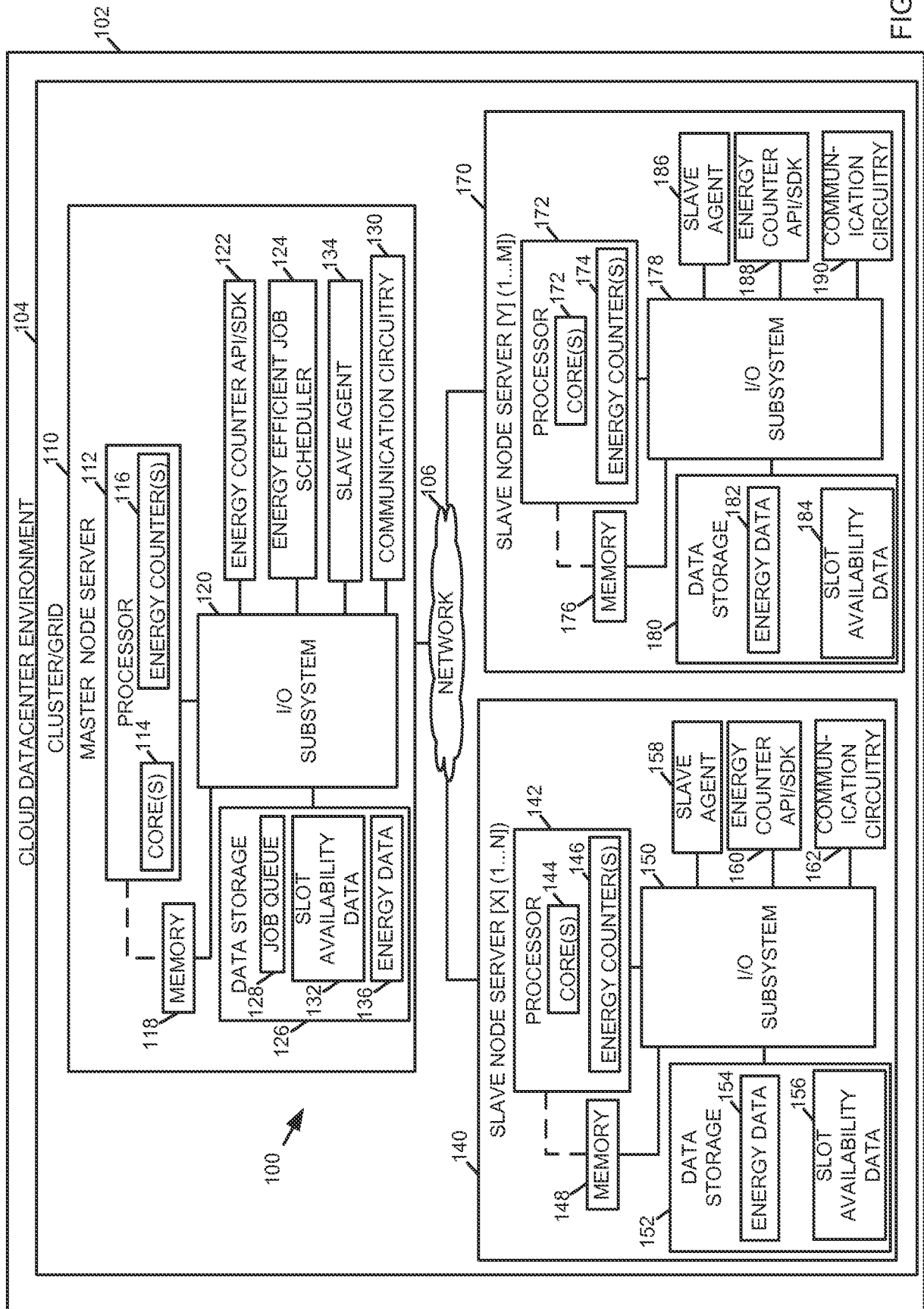


FIG. 1

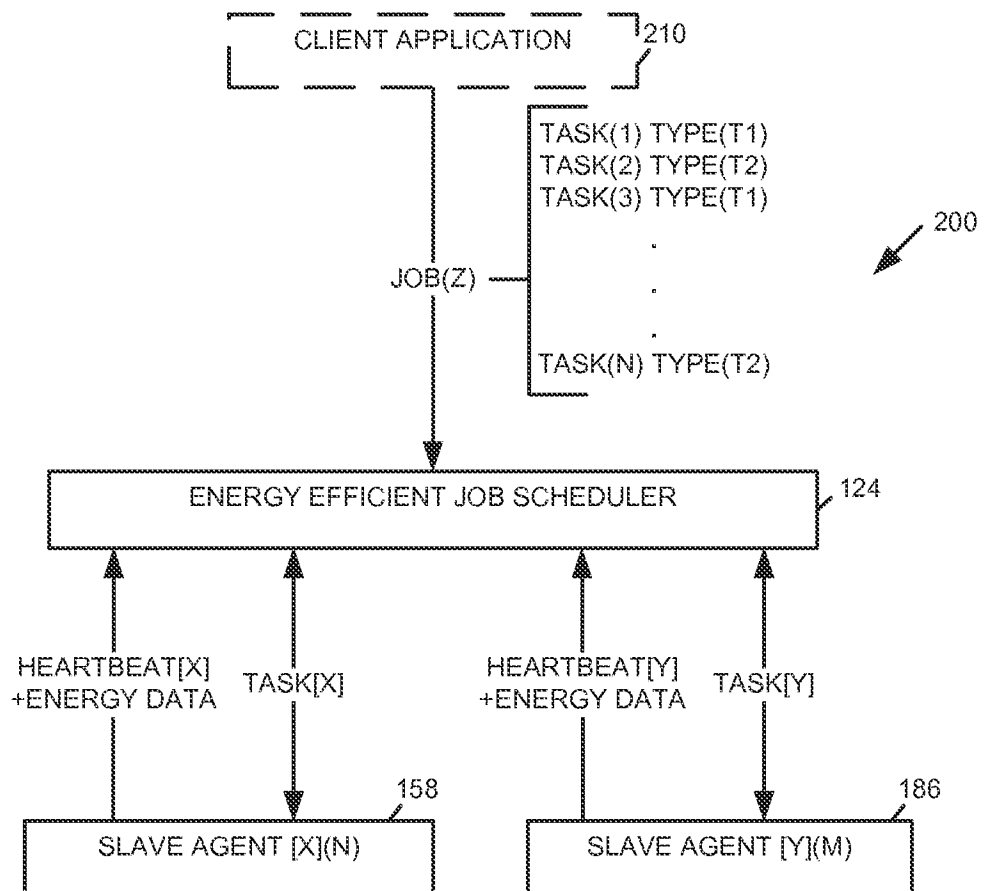
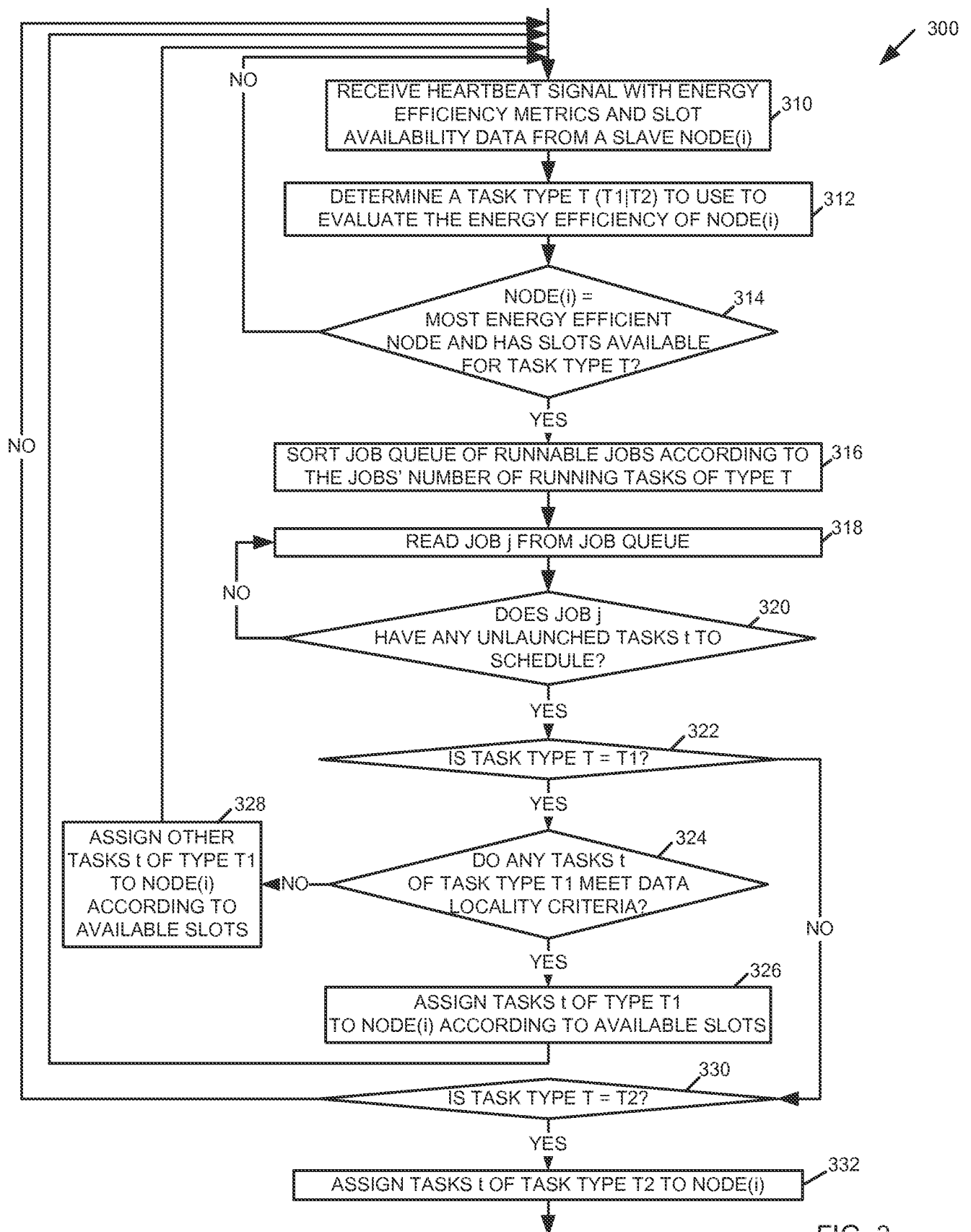


FIG. 2



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/045990**A. CLASSIFICATION OF SUBJECT MATTER****G06F 9/46(2006.01)i, G06F 15/16(2006.01)i**

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 9/46; G05B 19/418; H02J 3/14; G06F 1/26; H04L 12/28; G05D 23/19; G06F 15/16

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: cloud computing, server, node, energy, power, schedule

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2011-0161696 A1 (FLETCHER JAMES C.) 30 June 2011 See figure 1; paragraphs [3]-[5], [15]-[17] and [24]	1-5, 9-12, 15, 21
A	See figure 1; abstract	6-8, 13-14, 16-20, 22-26
Y	WO 2011-106917 A1 (LONGDHUA (BEIJING) CONTROLS TECHNOLOGY CO., LTD. et al.) 09 September 2011 See figure 1; pages 1, 3 and 4	1-5, 9-12, 15, 21
A	See figure 1; abstract	6-8, 13-14, 16-20, 22-26
A	WO 2011-011404 A1 (ALLURE ENERGY, INC. et al.) 27 January 2011 See figure 6; paragraph [172]	1-26
A	US 2011-0283119 A1 (SZU HSING CHUNG et al.) 17 November 2011 See figure 1; abstract	1-26



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 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

17 October 2013 (17.10.2013)

Date of mailing of the international search report

17 October 2013 (17.10.2013)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City,
302-701, Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

YOON Hye Sook

Telephone No. +82-42-481-8370



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2013/045990

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0161696 A1	30/06/2011	US 8341441 B2	25/12/2012
WO 2011-106917 A1	09/09/2011	CN102193528 A	21/09/2011
		CN102193528 B	14/08/2013
		EP 2544059 A1	09/01/2013
		JP 2013-523060A	13/06/2013
		US 2013-0066477 A1	14/03/2013
WO 2011-011404 A1	27/01/2011	AU 2010-276364 A1	16/02/2012
		AU 2010-276364 A2	23/08/2012
		CA 2768799 A1	27/01/2011
		CN102498448 A	13/06/2012
		EP 2457132 A1	30/05/2012
		JP 2012-533727A	27/12/2012
		KR20120087887A	07/08/2012
		MX2012000906 A	07/09/2012
		US 2010-0076613 A1	25/03/2010
		US 2011-0015802 A1	20/01/2011
		US 2011-0016018 A1	20/01/2011
		US 2011-0046792 A1	24/02/2011
		US 2011-0046798 A1	24/02/2011
		US 2011-0046799 A1	24/02/2011
		US 2011-0046800 A1	24/02/2011
		US 2011-0046801 A1	24/02/2011
		US 2011-0051823 A1	03/03/2011
		US 2011-0054699 A1	03/03/2011
		US 2011-0054710 A1	03/03/2011
		US 2011-0172835 A1	14/07/2011
		US 2011-0173542 A1	14/07/2011
		US 2011-0214060 A1	01/09/2011
		US 2011-0246898 A1	06/10/2011
		US 2012-0023225 A1	26/01/2012
		US 2012-0046859 A1	23/02/2012
		US 2012-0064923 A1	15/03/2012
		US 2012-0135759 A1	31/05/2012
		US 2012-0252430 A1	04/10/2012
		US 2013-0167035 A1	27/06/2013
		US 7930070 B2	19/04/2011
		US 8024073 B2	20/09/2011
		US 8082065 B2	20/12/2011
		US 8099195 B2	17/01/2012
		US 8108076 B2	31/01/2012
		US 8174381 B2	08/05/2012
		US 8265798 B2	11/09/2012
		US 8396602 B2	12/03/2013
		US 8396604 B2	12/03/2013
		US 8428782 B2	23/04/2013
		US 8457797 B2	04/06/2013
		US 8498749 B2	30/07/2013

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2013/045990

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0283119 A1	17/11/2011	US 8504180 B2	06/08/2013
		US 8509954 B2	13/08/2013
		CN103138370 A	05/06/2013
		TW201211749 A	16/03/2012