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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 August 2016 (04.08.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/122988 A2

(51) International Patent Classification:
G06F 1/32 (2006.01)

(21) International Application Number:
PCT/US2016/014628

(22) International Filing Date:
22 January 2016 (22.01.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/108,033 26 January 2015 (26.01.2015) US
15/003,551 21 January 2016 (21.01.2016) US

(71) Applicant: **APPLE INC.** [US/US]; 1 Infinite Loop, Cupertino, California 95014 (US).

(72) Inventor: **KUMAR, Derek R.**; 1 Infinite Loop, Mail Stop 301-2COS, Cupertino, California 95014 (US).

(74) Agents: **SCHELLER, James C.** et al.; Blakely Sokoloff Taylor & Zafman LLP, 1279 Oakmead Parkway, Sunnyvale, California 94085 (US).

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

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

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR SOC IDLE POWER STATE CONTROL BASED ON I/O OPERATION CHARACTERIZATION

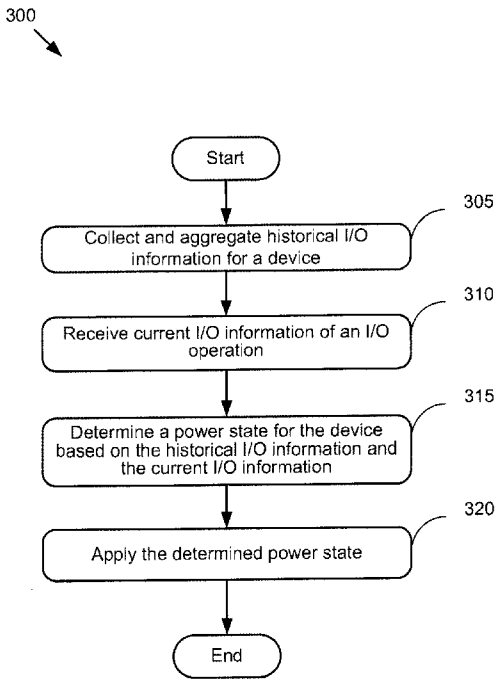

FIG. 3A

(57) Abstract: A method and apparatus of a device that manages system performance by controlling power state based on information related to I/O operations is described. The device collects historical I/O information. The historical I/O information may include the number of I/O operations over a sample period of time and the inter-arrival time between I/O operations. The device further receives information related to a current I/O operation. The information of the current I/O operation may include direction, size, quality of service, and media type of the I/O operation. The device determines a power state based on the historical I/O information and the information relative to the current I/O operation to reduce power consumption while improving system efficiency and maintaining an acceptable level of system performance. The device further applies the determined power state. Other embodiments are also described and claimed.

WO 2016/122988 A2

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

SYSTEM AND METHOD FOR SOC IDLE POWER STATE CONTROL BASED ON I/O OPERATION CHARACTERIZATION

Related Matters

[0001] This application claims the benefit of the earlier filing date of provisional application no. 62/108,033, filed January 26, 2015, entitled "System and Method for SoC Idle Power State Control Based on I/O Operation Characterization".

Field of the Disclosure

[0002] This disclosure relates generally to operating systems and more particularly to devices for performance management.

BACKGROUND OF THE DISCLOSURE

[0003] An operating system is a collection of software that manages device hardware resources and provides common services for computer programs. The operating system is a vital component of the system software in a device. The system software manages and integrates a device's capabilities. The system software includes the operating system, utility software, device drivers, and other software tools. The operating system manages the power consumption and performance of the device.

[0004] Power Management for a device power gates a processor to a low-power state when the processor is inactive. The low-power state can be an idle state for the processor. Modern processors within computer systems often adopt aggressive power management techniques in order to reduce overall energy consumption, reduce cooling requirements, and prolong battery life for portable and embedded systems. As a result, the processors enter into low-power states more frequently and dwell in such low-power states for longer periods of time. Transitions to these low power states typically involves additional latency, with frequent transitions to/from such states reducing net efficiency gains. The processor may support multiple types of idle power states, with each involving differing transition latencies, transition energies, and achievable power levels.

[0005] Input/output (I/O) is the communication between a computer and the outside world. Inputs are the signals or data received by the computer and outputs are the signals or data sent from it. Any transfer of information to or from the CPU or memory of a computer, for example by reading data from a disk drive, is considered I/O.

SUMMARY

[0006] A method and apparatus of a device that manages system performance by controlling power state based on I/O information is described. In one embodiment, the I/O information is the current and historical information related to I/O operations. In one embodiment, the current I/O information includes the direction (*e.g.*, read or write) of the current I/O operation. In one embodiment, the current I/O information includes the size (*e.g.*, the amount of data involved) of the current I/O operation. In one embodiment, the current I/O information includes the quality of service of the current I/O operation. In one embodiment, the current I/O information includes the media type (*e.g.*, solid state drive or hard drive) of the current I/O operation. The historical I/O information of one embodiment includes the number of I/O operations over a sample period of time. In one embodiment, the historical I/O information includes the inter-arrival time between I/O operations over a sample period of time. In one embodiment, the historical I/O information includes the aggregated size of read operations over a sample period of time and the aggregated size of write operations over the same sample period of time.

[0007] In an exemplary embodiment, the device collects historical I/O information. The device further receives current I/O information of an I/O operation to be performed by the device. The device determines a power state based on the historical I/O information and the current I/O information to reduce power consumption while improving system efficiency and maintaining an acceptable level of system performance. The device further applies the determined power state. In one embodiment, the power state is applied to the entire device. In another embodiment, the power state is applied to a processor of the device. In yet another embodiment, the power state is applied to a system on chip (SoC) of the device.

[0008] Other methods and apparatuses are also described. Machine-readable non-transitory media are also described and they include executable computer program instructions which when executed by a data processing system cause the data processing system to perform one or more methods described herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The present disclosure is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

[0010] **Figure 1** is a block diagram of one embodiment of a device that manages system performance by controlling power state based on I/O information.

[0011] **Figure 2** illustrates a detailed block diagram of one embodiment of a power manager.

[0012] **Figure 3A** illustrates a flowchart of one embodiment of a process to manage system performance by controlling power state based on I/O information.

[0013] **Figure 3B** illustrates a flowchart of one embodiment of another process to manage system performance by controlling power state based on I/O information.

[0014] **Figure 4** illustrates an example of current I/O information used in one embodiment.

[0015] **Figure 5** illustrates an example of historical I/O information used in one embodiment.

[0016] **Figure 6** illustrates an example in one embodiment for adjusting power state based on I/O information.

[0017] **Figure 7** illustrates one example of a data processing system, which may be used with one embodiment of the present disclosure.

[0018] **Figure 8** illustrates one example of another data processing system, which may be used with one embodiment of the present disclosure.

DETAILED DESCRIPTION

[0019] A method and apparatus of a device that manages system performance by controlling power state based on I/O information is described. In the following description, numerous specific details are set forth to provide thorough explanation of embodiments of the present disclosure. It will be apparent, however, to one skilled in the art, that embodiments of the present disclosure may be practiced without these specific details. In other instances, well-known components, structures, and techniques have not been shown in detail in order not to obscure the understanding of this description.

[0020] Reference in the Specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment can be included in at least one embodiment of the disclosure. The appearances of the phrase "in one embodiment" in various places in the Specification do not necessarily all refer to the same embodiment.

[0021] In the following description and claims, the terms "coupled" and "connected," along with their derivatives, may be used. It should be understood that these terms are not intended as synonyms for each other. "Coupled" is used to indicate that two or more elements, which may or may not be in direct physical or electrical contact with each other, co-operate or interact with each other. "Connected" is used to indicate the establishment of communication between two or more elements that are coupled with each other.

[0022] The processes depicted in the figures that follow, are performed by processing logic that comprises hardware (*e.g.*, circuitry, dedicated logic, etc.), software (such as is run on a general-purpose device or a dedicated machine), or a combination of both. Although the processes are described below in terms of some sequential operations, it should be appreciated

that some of the operations described may be performed in different order. Moreover, some operations may be performed in parallel rather than sequentially.

[0023] The terms "server," "client," and "device" are intended to refer generally to data processing systems rather than specifically to a particular form factor for the server, client, and/or device.

[0024] At times, after a processor performs an I/O operation (e.g., issuing a read/write command to a NAND storage device), the processor and/or system on a chip (SoC), in the absence of other pending work, can enter a low power state. In one embodiment, this low-power state is an idle state. In this embodiment, a power management module causes the processor, associated caches and memory controllers, and random access memory (RAM) to enter low power states. Depending on the power state selected, processor caches may be evicted, and RAM may enter self-refresh. Not long after that, the NAND storage device completes the I/O request, and the memory controller and the RAM are reactivated in order to access the data. An interrupt is generated to notify the processor of the completion of the I/O and restore the processor and associated components from the low-power state to a full-power state so that the processor can operate on the data returned by the I/O operation.

[0025] Because it takes time to drag the processor and related components out of the low-power state after the I/O request is completed, there is latency in receiving the data from the I/O subsystem. Therefore, it will take an increased amount of time for the I/O operation to complete. That might be unacceptable because the user is waiting for the data to be dragged into RAM, possibly during an application launch or when the device is paging in memory stored on the storage device. So there is negative impact to the performance of the computer system by frequently having the processor and related components entering into the low-power state after issuing I/O requests.

[0026] Furthermore, by having the processor and related components entering into the low-power state, just to have them immediately back to a normal power state, a significant amount of energy is wasted to perform the transitions between different power states, including power gating the processor and related components down to enter into the low-power state and then immediately reactivating those components to exit the lower power state. So there is also negative impact to the efficiency of the computer system.

[0027] A method and apparatus of a device that manages system performance in order to reduce power consumption of the device while improving system efficiency and maintaining a reasonable level of system performance is described. In one embodiment, after a processor issues an I/O request to a non-volatile storage device, the processor enters into a low-power state under which power to the processor and related components are power gated to reduce

power consumption of the device. The non-volatile storage device completes the I/O transaction and notifies the processor to further process the data involved in the I/O transaction. The processor and related components are reactivated to process the data. This introduces latency in receiving the data from the I/O subsystem and adds unnecessary energy cost of entering and exiting the low-power state. Latency is the amount of time it takes to go from a low-power state to a higher power state in which the processor can perform certain intended functions. In one embodiment, the device determines a power state based on the historical I/O information and the current I/O information to reduce power consumption while improving system efficiency and maintaining an acceptable level of system performance. The device then enters into the determined power state. In one embodiment, the device applies the determined power state when the I/O operation is in flight. In another embodiment, the device applies the determined power state when the I/O operation is in flight and after receiving the result of the I/O operation.

[0028] **Figure 1** is a block diagram of one embodiment of a device 100 that manages system performance by controlling power state based on I/O information. In one embodiment, the device 100 can be a desktop computer, server, smartphone, laptop, personal digital assistant, music playing device, gaming device, or any other device that can execute multiple processes. In **Figure 1**, device 100 includes an integrated circuit 110, an operating system 102, a volatile memory 130, and a non-volatile memory 135. In one embodiment, the device 100 can include multiple processors, and/or multiple processing cores.

[0029] The integrated circuit 110 includes a processor 120 and a memory controller 125. In one embodiment, the integrated circuit 110 can be a system on chip (SoC). The processor 120 is a multipurpose, programmable device that accepts digital data as input, processes the data according to instructions stored in its memory, and provides results as output. The processor 120 includes a block storage driver 122 that controls the block storage devices (*e.g.*, non-volatile memory 135) that are attached to the processor 120. The memory controller 125 manages the flow of data going to and from the volatile memory 130.

[0030] The operating system 102 is a set of software used to manage device hardware resources and provides common services for other running computer programs, such as application programs. The operating system 102 includes a power manager 105 that manages power consumption of the device 100. The power manager 105 can determine a power state for the entire device 100, or power states for individual components of the device. In one embodiment, the power manager 105 applies a power state to the integrated circuit 110. In another embodiment, the power manager 105 applies a power state to the processor 120.

[0031] A power state defines a power consumption level of a device, or one or more components of the device. For example and in one embodiment, a power state can be a full-

power state (*i.e.*, the highest power state) or a power-off state (*i.e.*, the lowest power state). Under the full-power state, the device or the group of components of the device are fully powered on, with no individual components set in a power saving mode. Under the power-off state, the device or components are turned off. In another embodiment, there can be several low-power states, which define different power consumption levels that are between the full-power state and the power-off state. For a device or a group of components, the more components are in the power saving mode, the lower the power state is.

[0032] For components that are in power saving mode, different power reduction techniques can define different power states. For example, power gating is a power reduction technique that shuts off the current to blocks of the circuit to reduce power consumption. Clock gating is a power reduction technique that saves power by disabling clock pulses to portions of a circuit so that the transistors in those portions of the circuit do not switch states. Even though the power consumption for switching states is avoided, leakage currents are still incurred in clock gating. Therefore, power gating saves more power than clock gating, and components that are power gated are in a lower power states than components that are clock gated. However, power-gating may not retain state, such as the contents of processor caches, which may need to be reconstructed on power-gate exit.

[0033] The non-volatile memory 135 is a block storage device that persistently stores information, such as a hard disk or a NAND stack. The volatile memory 130 is computer memory that requires power to maintain the stored information, such as dynamic random-access memory (DRAM).

[0034] In one embodiment, the block storage driver 122 issues read/write commands 123 (by issuing I/O requests) to access data on the non-volatile memory 135. In the cases of read commands, the non-volatile memory 135 sends the requested data 136 to the memory controller 125, which passes the data 136 to the volatile memory 130 for further processing.

[0035] In one embodiment, while sending the read/write commands 123, the block storage driver 122 also sends I/O information related to the current I/O request to the power manager 105. In one embodiment, the power manager 105 determines a low-power state based on the current I/O information 106 and stored historical I/O information (not shown) in order to reduce power consumption while improving efficiency and maintaining an acceptable level of system performance. For example and in one embodiment, instead of dropping the processor 120 and its related components to a very low-power state under which the processor 120 is power gated (e.g., a power-off state), the power manager 105 determines, based on the current I/O information 106 and historical I/O information (not shown), a low-power state under which the processor is clock gated, thus having a higher level of power consumption. By determining a

higher power state and applying the higher power state to the processor 120, the latency and negative efficiency impact caused by entering into the very low power state and then quickly exiting it during certain I/O operations can be avoided.

[0036] In one embodiment, the current I/O information 106 may include direction, size, quality of service (QoS), and media type of the current I/O operation. In one embodiment, the power manager 105 applies the determined power state to the entire device 100. In another embodiment, the power manager 105 applies the determined power state to the integrated circuit 110. In yet another embodiment, the power manager 105 applies the determined power state to the processor 120 and a few related components, such as memory controller 125 and volatile memory 130.

[0037] In one embodiment, the integrated circuit 110 includes multiple processors, and/or multiple processing cores. In that embodiment, the power manager 105 controls the idle power state of the entire integrated circuit 110 by applying the determined power state to the integrated circuit 110. The integrated circuit 110 is constrained on the idle state it can achieve, even if one core is active. For example and in one embodiment, the power manager 105 controls the idle power state of the core receiving completion interrupts from the I/O subsystem by applying the determined power state to that core. In one embodiment, the completion interrupts from the I/O subsystem are restricted to be receivable by a designated core to avoid complexity in implementation.

[0038] **Figure 2** illustrates a detailed block diagram of one embodiment of a power manager 105. In one embodiment, the power manger 105 is part of an operating system, as described in relation to **Figure 1** above. In another embodiment, the power manager 105 is a separate hardware/software module that manages power consumption of a device. In **Figure 2**, the power manager 105 includes an integrator 210 and a historical data storage 220.

[0039] In one embodiment, the integrator 210 receives current I/O information 106 from a block storage driver (not shown), and stores the current I/O information 106 into the historical I/O data storage 220. In one embodiment, the current I/O information 106 may include direction, size, quality of service, and media type of the current I/O operation. The integrator 210 also receives historical I/O data 225 from the historical data storage 220. In one embodiment, the historical I/O data 225 may include the number of I/O operations over a sample period of time, the inter-arrival time between I/O operations, the aggregated size of read operations, and the aggregated size of write operations. In one embodiment, the integrator 210 determines a power state 230 based on the current I/O information 106 and the historical I/O data 225.

[0040] In one embodiment, the power state 230 outputted by the integrator 210 is a low-power state. In one embodiment, the determined power state 230 is applied to an entire device. In another embodiment, the determined power state 230 is applied to a processor and components closely associated with the processor, *e.g.* memory controller and DRAM.

[0041] The historical I/O data storage 220 receives current I/O information 106 and stores the current the current I/O information 106 as part of the historical I/O data. In one embodiment, the information for each I/O operation is stored as a separate item in the historical I/O data storage 220. In another embodiment, the information for I/O operations are aggregated and stored in the historical I/O data storage 220. The historical I/O data storage 220 provides the historical I/O data 225 to the integrator 210.

[0042] In one embodiment, the integrator 210 determines an appropriate low-power state 230 based on the current I/O information 106 and the historical I/O data 225 in order to reduce power consumption while improving efficiency and maintaining an acceptable level of system performance. For example in one embodiment, instead of dropping a processor to a very low-power state under which most circuit blocks of the processor are power gated, the integrator 210 selects, based on the current I/O information 106 and historical I/O data 225, a power state 230 under which most circuit blocks of the processor are clock gated, which incurs more energy consumption, but involves lower latency transitions. By selecting this higher power state and applying the higher power state to the processor, rather than re-initializing and reloading all state as it would be when most of the processor is power gated, the clock gated processor just receives a signal to disable the clock gating when the interrupt indicating an I/O operation has completed arrives. Consequently, the latency and negative efficiency impact caused by entering into the very low-power state and then quickly exiting it during certain I/O operations can be avoided.

[0043] For example, paging is a memory management mechanism in which a computer stores and retrieves data in same-size blocks (i.e., pages) from the secondary storage for use in main memory. During a paging storm, a lot of paging requests are issued and a large number of I/O operations are performed in a short period of time. For example and in one embodiment, a paging storm is generated when an application program is being launched. When power state control is not based on I/O information, each time an I/O request is issued during the paging storm, the processor may be dropped into a deep low-power state (*e.g.,* a power gated state that has a very low level of power consumption and would have a high latency in order to be re-activated). Once the I/O operation is completed, it will take a long time and consume quite a bit of power in order to drag the processor out of the deep low-power state to further process the data received from the I/O subsystem. This would have a negative impact on the system

performance and efficiency because the user would have to wait for a longer period of time for the application to be launched and power is wasted in frequent entering and exiting deep low-power state.

[0044] With I/O based power state control, each time an I/O request is issued during the paging storm, the power state of the processor is determined by taking into account the characteristics of the I/O operation and historical I/O data. In the scenario of a paging storm caused by an application launch, each I/O operation has a high QoS because the user is waiting for the application program to be launched. The historical I/O information related to the paging storm would indicate that the number of I/O operations is large and the inter-arrival time between I/O operations is short. Consequently, a shallow low-power state (*e.g.,* an idle processor power state of relatively high power consumption level and very low latency) will be selected across I/O operations involved in the paging storm to save a small amount of power while improving system efficiency and maintaining a high level of system performance. The user-perceivable wait for the application to be launched would be shorter and less energy is being consumed in the transition between different power states.

[0045] The power manager 105 was described above for one embodiment of the disclosure. One of ordinary skill in the art will realize that in other embodiments this module can be implemented differently. For instance, in one embodiment described above, certain modules are implemented as software modules. However, in another embodiment, some or all of the modules might be implemented by hardware, which can be dedicated application specific hardware (*e.g.*, an ASIC chip or component) or a general purpose chip (*e.g.*, a microprocessor or FPGA).

[0046] **Figure 3A** illustrates a flowchart of one embodiment of a process 300 to manage system performance by controlling power state based on I/O information. In one embodiment, the power manger 105 described in **Figure 1** and **Figure 2** above executes process 300 to determine an appropriate low-power state in order to reduce power consumption while improving system efficiency and maintaining an acceptable level of system performance. In **Figure 3A**, process 300 begins by collecting and aggregating (at block 305) historical I/O information for a device. In one embodiment, the integrator 210 collects current I/O information and aggregates the current I/O information into historical I/O information, as described above in **Figure 2**. In one embodiment, the historical I/O information may include the number of I/O operations over a sample period of time, the inter-arrival time between I/O operations, the aggregated size of read operations, and the aggregated size of write operations.

[0047] At block 310, process 300 receives I/O information regarding a current I/O operation. In one embodiment, the current I/O information may include direction, size, quality of service, and media type of the current I/O operation.

[0048] In one embodiment, the current I/O information is information about the I/O operation to be performed. In another embodiment, the current I/O information is predictive information. For example and in one embodiment, the operating system may notice sequential accesses to a file. The system may choose to prepare for the I/O operation but not actually issue an I/O request for the I/O operation, and set a flag to indicate that this I/O operation is predictive.

[0049] At block 315, process 300 determines a power state for the device based on the historical I/O information and the current I/O information. In one embodiment, the power state determined at block 315 is a low-power state that reduces power consumption while improving system efficiency and maintaining a reasonable level of system performance. For example and in one embodiment, instead of dropping a processor to a very low-power state under which the processor is power gated, process 300 determines, based on the current I/O information and historical I/O information, a power state under which the processor is clock gated, thus having a higher level of power consumption. By determining a higher power state and applying the higher power state to the processor, the latency and negative efficiency impact caused by entering into the very low-power state and then quickly exiting it during certain I/O operations can be avoided.

[0050] In one embodiment, when the current I/O is flagged to be predictive, the determination of the power state can be planned in longer term. For example and in one embodiment, process 300 caps the total latency for a series of I/O operations and determines the power state accordingly. In one embodiment, process 300 may reduce the aggressiveness in power management when the current I/O is flagged to be predictive.

[0051] Process 300 applies (at block 320) the determined power state. In one embodiment, the power state is applied to the entire device. In another embodiment, the power state is applied to a processor of the device and components related to the processor, such as memory controller and DRAM.

[0052] One of ordinary skill in the art will recognize that process 300 is a conceptual representation of the operations used to manage system performance. The specific operations of process 300 may not be performed in the exact order shown and described. The specific operations may not be performed in one continuous series of operations, and different specific operations may be performed in different embodiments. Furthermore, process 300 could be implemented using several sub-processes, or as part of a larger macro process.

[0053] The process of I/O based power state control is described above in **Figure 3A** from the power manager's perspective. **Figure 3B** illustrates a flowchart of one embodiment of another process 350 to manage system performance by controlling power state based on I/O information. **Figure 3B** describes the process of I/O based power state control from the I/O operation's perspective. In one embodiment, the device 100 described in **Figure 1** above executes process 350. In **Figure 3B**, process 350 begins by collecting (at block 355) historical I/O data. In one embodiment, the historical I/O data may include the number of I/O operations over a sample period of time, the inter-arrival time between I/O operations, the aggregated size of read operations, and the aggregated size of write operations.

[0054] At block 360, process 350 determines an I/O operation to be performed. In one embodiment, the I/O operation may be a read operation or a write operation. At block 365, process 350 determines characteristics of the I/O operation. In one embodiment, the characteristics of the I/O operation may include direction, size, quality of service, and media type of the I/O operation.

[0055] In one embodiment, the current I/O information is information about the I/O operation to be performed. In another embodiment, the current I/O information is predictive information. For example and in one embodiment, the operating system may notice sequential accesses to a file. The system may choose to prepare for the I/O operation but not actually issue an I/O request for the I/O operation, and set a flag to indicate that this I/O operation is predictive.

[0056] At block 370, process 350 determines a low-power state based on the historical I/O data and the characteristics of the I/O operation. In one embodiment, the low-power state determined at block 370 is a low-power state that reduces power consumption while improving system efficiency and maintaining a reasonable level of system performance. For example and in one embodiment, instead of dropping a processor to a very low-power state under which the processor is power gated, process 350 determines, based on the characteristics of the I/O operation and historical I/O data, a low-power state under which the processor is clock gated, thus having a higher level of power consumption. By determining a higher power state and applying the higher power state to the processor, the latency and negative efficiency impact caused by entering into the very low-power state and then quickly exiting it during certain I/O operations can be avoided.

[0057] In one embodiment, when the current I/O is flagged to be predictive, the determination of the low-power state can be planned in longer term. For example and in one embodiment, process 350 caps the total latency for a series of I/O operations and determines the power state accordingly. In one embodiment, process 350 may reduce the aggressiveness in power management when the current I/O operation is flagged to be predictive. For example and in one

embodiment, a less aggressive power management is implemented by using a clock-gated state instead of a power gated state.

[0058] At block 375, process 350 issues an I/O request for the I/O operation. Process 350 enters (at block 380) the determined low-power state. In one embodiment, the low-power state is applied to the entire device. In another embodiment, the low-power state is applied to a processor of the device and components related to the processor, such as memory controller and DRAM.

[0059] At block 385, process 350 receives a notification that the I/O operation has completed. In one embodiment, the notification is received from the storage device that performs the I/O operation. At block 390, the processor comes out of low-power state to process result of the I/O operation.

[0060] One of ordinary skill in the art will recognize that process 350 is a conceptual representation of the operations used to manage system performance. The specific operations of process 350 may not be performed in the exact order shown and described. For example, the operations of blocks 370 can 375 can be performed in parallel or in reverse order. The specific operations may not be performed in one continuous series of operations, and different specific operations may be performed in different embodiments. Furthermore, process 350 could be implemented using several sub-processes, or as part of a larger macro process.

[0061] **Figure 4** illustrates an example of current I/O information used in one embodiment. As illustrated, the current I/O information 106 contains information items that include, but not limited to, I/O direction 405, I/O size 410, I/O QoS 415, and I/O media type 420. In one embodiment, one or more of these information items for the current I/O operation may be used in the determination of an appropriate low-power state in order to reduce power consumption while improving system efficiency and maintaining an acceptable level of system performance.

[0062] The I/O direction 405 represents the direction (*e.g.*, read or write) of the I/O operation. In one embodiment, if the I/O operation is a write operation, the current I/O information 106 is ignored in the determination of the low-power state because a write operation may be buffered at the block storage and the processor does not act on the completion of the write operation. In that embodiment, information about a read operation may impact the determination of power state, while information about a write operation does not impact the determination of power state. In one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and lower latency than it would be without considering the I/O information) will be selected while information about a read operation is received.

[0063] A person of ordinary skill in the art will realize that high power consumption level is less than the full operating power level, and low power consumption level is more than

completely powering off. A person of ordinary skill in the art will also realize that "high" is relative to "no power" and "low" is relative to "full power".

[0064] The I/O size 410 represents the size (*e.g.*, the amount of data involved) of the I/O operation. In one embodiment, a large size I/O operation is given less weight in the determination of the low-power state than a small size I/O operation because a large size I/O operation takes longer time to complete and has less impact on the immediate future of the power state. In contrast, a small size I/O operation is completed quickly, thus has more impact on the immediate future of the power state. In one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and lower latency) will be selected for a small size I/O operation, while a deeper low-power state (*i.e.*, a power state of lower power consumption level and higher latency) will be selected for a large size I/O operation.

[0065] The I/O QoS 415 represents whether the I/O operation is important. In one embodiment, an I/O operation marked with low QoS has little or no impact on the determination of power state. In that embodiment, an I/O operation that marked with high QoS (*e.g.*, a paging operation) has more impact on the determination of power state. For example and in one embodiment, an I/O operation with low QoS can be either filtered out completely or weighted lower in the determination of power state. For example and in one embodiment, an I/O operation with median QoS might be weighted just half as much as an important I/O operation (*e.g.*, from paging) in the determination of power state. For example and in one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and lower latency) will be selected for a I/O operation with high QoS, while a deeper low-power state (*e.g.*, a power state about half of the power consumption level of the power state for the I/O operation with high QoS) will be selected for an I/O operation with median QoS.

[0066] The I/O media type 420 represents the type of media (*e.g.*, solid state drive or hard drive) on which the I/O operation is performed. In one embodiment, an I/O operation performed on a hard drive has less impact on the determination of power state than an I/O operation performed on a solid state drive, because the latency for an I/O operation on hard drive is longer, thus the completion of the I/O operation has less impact on the immediate future of power state. In one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and lower latency) will be selected for an I/O operation on solid state drive, while a deeper low-power state (*i.e.*, a power state of lower power consumption level and higher latency) will be selected for an I/O operation on a hard drive.

[0067] The effect of each information item of the current I/O information 106 to the determination of low-power state has been discussed separately above. However, a person skilled in the art will recognize that two or more information items can be combined to further

improve the determination of low-power state. In one embodiment, a low-power state is determined based on the cumulative effect of two or more information items of the current I/O information 106.

[0068] **Figure 5** illustrates an example of historical I/O information used in one embodiment. As illustrated, the historical I/O data 225 contains information items that include, but not limited to, the number of I/O operations 505, inter-arrival time between I/O operations 510, aggregated size of read operations 515, and aggregated size of write operations 520. In one embodiment, one or more of these information items for the historical I/O data may be used in the determination of an appropriate low-power state in order to reduce power consumption while improving system efficiency and maintaining an reasonable level of system performance.

[0069] The number of I/O operations 505 represents a count of I/O operations during a sample period of time. In one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and latency) will be selected when the number of I/O operations is high, while a deeper low-power state (*i.e.*, a power state of lower power consumption level and higher latency) will be selected when the number of I/O operations is low. In one embodiment, the higher the number of I/O operations is, the shallower the low-power state will be.

[0070] In one embodiment, the inter-arrival time between I/O operations represents the average inter-arrival time between I/O operations during a sample period of time. In one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and low latency) will be selected if the inter-arrival time between I/O operations is short, while a deeper low-power state (*e.g.*, a power state of lower power consumption level and higher latency) will be selected if the inter-arrival time between I/O operations is long. In one embodiment, the shorter the inter-arrival time between I/O operations is, the shallower the low-power state will be.

[0071] The aggregated size of read operations 515 represents the total size of read operations during a sample period of time. In one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and low latency) will be selected if the aggregated size of read operations is small, while a deeper low-power state (*e.g.*, a power state of lower power consumption level and higher latency) will be selected if the aggregated size of read operations is large. In one embodiment, the smaller the aggregated size of read operations is, the shallower the low-power state will be.

[0072] The aggregated size of write operations 520 represents the total size of write operations during a sample period of time. In one embodiment, a shallower low-power state (*i.e.*, a power state of higher power consumption level and lower latency) will be selected if the aggregated size of write operations is small, while a deeper low-power state (*e.g.*, a power state

of lower power consumption level and higher latency) will be selected if the aggregated size of write operations is large. In one embodiment, the smaller the aggregated size of write operations is, the shallower the low-power state will be.

[0073] The effect of each information item of the historical I/O data 225 to the determination of low-power state has been discussed separately above. However, a person skilled in the art will recognize that two or more information items can be combined to further improve the determination of low-power state. In one embodiment, a low-power state is determined based on the cumulative effect of two or more information items of the historical I/O data 225.

[0074] **Figure 6** illustrates an example in one embodiment for adjusting power state based on I/O information. **Figure 6** identifies several different points of time 610A-F on *x*-axis of chart 600. Three I/O operations happened during this period of time. The first I/O operation is issued at time 610A and completed at time 610B. The second I/O operation is issued at time 610C and completed at time 610D. The third I/O operation is issued at time 610E and completed at time 610F.

[0075] **Figure 6** also identifies several different power states 620A-E on *y*-axis of chart 600. Latency is the amount of time it takes to go from a low-power state to a higher power state in which the processor can perform certain intended functions. The closer the power state is to the origin of chart 600, the lower power consumption level and higher latency the power state has. The farther away the power state is to the origin of chart 600, the higher power consumption level and lower latency the power state has. For example, power state 620D has a higher power consumption level and lower latency than power states 620A-C.

[0076] In one embodiment, the first, second, and third I/O operations are of the same characteristics, *e.g.* the same size and QoS, etc. In that embodiment, the power state is set at 620A before the first I/O operation is issued. At time 610A, there is no historical I/O information. In one embodiment, no historical I/O information means there is no historical I/O information stored. In another embodiment, no historical I/O information means the older historical I/O information has become stale and is not used for the analysis. In yet another embodiment, no historical I/O information means there are no I/O operations within a time period. At time 610A, the power state is raised to 620B to issue the first I/O operation. At time 610B, the first I/O operation is completed and the power state is dropped to 620B.

[0077] At time 610C, there is historical I/O information available with the happening of the first I/O operation and possibly other I/O operations before that time. Because there is I/O historical information available, the power state is raised to 620D at time 610C, which is a bigger raise of power consumption level than at time 610A, to issue the second I/O operation. At time 610D, the second I/O operation is completed and power state is dropped to 620C, which

has higher power consumption level and lower latency than power state 620B after the first I/O operation is completed.

[0078] At time 610E, more historical I/O information is available with the happening of the first and second I/O operations and possibly other I/O operations before that time. Because there is more I/O historical information available, the power state is raised to 620E at time 610E, which is an even bigger raise of power consumption level than at time 610C, to issue the third I/O operation. At time 610F, the third I/O operation is completed and power state is dropped to 620D, which has even higher power consumption level and lower latency than power state 620C after the second I/O operation is completed.

[0079] **Figure 6** shows the accelerated rising of power state when more and more historical I/O information is available. So that a power state with high power consumption level and low latency is maintained to handle I/O operations to reduce power consumption while improving system efficiency and to ensure a reasonable level of system performance.

[0080] **Figure 7** shows one example of a data processing system 700, which may be used with one embodiment of the present disclosure. For example, the system 700 may be implemented with the device 100 as shown in **Figure 1**. Note that while **Figure 7** illustrates various components of a device, it is not intended to represent any particular architecture or manner of interconnecting the components as such details are not germane to the present disclosure. It will also be appreciated that network computers and other data processing systems or other consumer electronic devices, which have fewer components or perhaps more components, may also be used with the present disclosure.

[0081] As shown in **Figure 7**, the device 700, which is a form of a data processing system, includes a bus 703 which is coupled to a microprocessor(s) 705, a ROM (Read Only Memory) 707, volatile RAM 709, and a non-volatile memory 711. The microprocessor 705 is coupled to cache memory 704. Cache memory 704 may be volatile or non-volatile memory. The microprocessor 705 may retrieve the instructions from the memories 707, 709, 711, and 704, and execute the instructions to perform operations described above. The bus 703 interconnects these various components together and also interconnects these components 705, 707, 709, and 711 to a display controller and display device 713 and to peripheral devices such as input/output (I/O) devices 715, which may be mice, keyboards, modems, network interfaces, printers, and other devices which are well known in the art.

[0082] Typically, the input/output devices 715 are coupled to the system through input/output controllers 710. The volatile RAM (Random Access Memory) 709 is typically implemented as dynamic RAM (DRAM), which requires power continually in order to refresh or maintain the data in the memory.

[0083] The mass storage 711 is typically a magnetic hard drive or a magnetic optical drive or an optical drive or a DVD RAM or a flash memory or other types of memory systems, which maintain data (*e.g.*, large amounts of data) even after power is removed from the system. Typically, the mass storage 711 will also be a random access memory, although this is not required. While **Figure 7** shows that the mass storage 711 is a local device coupled directly to the rest of the components in the data processing system, it will be appreciated that the present disclosure may utilize a non-volatile memory which is remote from the system, such as a network storage device which is coupled to the data processing system through a network interface such as a modem, an Ethernet interface or a wireless network. The bus 703 may include one or more buses connected to each other through various bridges, controllers, and/or adapters as is well known in the art.

[0084] **Figure 8** shows an example of another data processing system 800 which may be used with one embodiment of the present disclosure. For example, system 800 may be implemented as a device 100 as shown in **Figure 1**. The data processing system 800 shown in **Figure 8** includes a processing system 811, which may be one or more microprocessors, or which may be a system on a chip integrated circuit, and the system also includes memory 801 for storing data and programs for execution by the processing system. The system 800 also includes an audio input/output subsystem 805, which may include a microphone and a speaker, for example, for playing back music or providing telephone functionality through the speaker and microphone.

[0085] A display controller and display device 809 provide a visual user interface for the user; this digital interface may include a graphical user interface which is similar to that shown on a Macintosh computer when running OS X operating system software, or Apple iPhone® when running the iOS operating system, etc. The system 800 also includes one or more wireless transceivers 803 to communicate with another data processing system, such as the system 800 of **Figure 8**. A wireless transceiver may be a WLAN transceiver, an infrared transceiver, a Bluetooth transceiver, and/or a wireless cellular telephony transceiver. It will be appreciated that additional components, not shown, may also be part of the system 800 in certain embodiments, and in certain embodiments fewer components than shown in **Figure 8** may also be used in a data processing system. The system 800 further includes one or more communication ports 817 to communicate with another data processing system, such as the system in **Figure 10**. The communication port may be a USB port, Firewire port, Bluetooth interface, etc.

[0086] The data processing system 800 also includes one or more input devices 813, which are provided to allow a user to provide input to the system. These input devices may be a keypad or a keyboard or a touch panel or a multi touch panel. The data processing system 800 also includes an optional input/output device 815 which may be a connector for a dock. It will

be appreciated that one or more buses, not shown, may be used to interconnect the various components as is well known in the art. The data processing system shown in **Figure 8** may be a handheld device or a personal digital assistant (PDA), or a cellular telephone with PDA like functionality, or a handheld device which includes a cellular telephone, or a media player, such as an iPod®, or devices which combine aspects or functions of these devices, such as a media player combined with a PDA and a cellular telephone in one device or an embedded device or other consumer electronic devices. In other embodiments, the data processing system 800 may be a network computer or an embedded processing device within another device, or other types of data processing systems, which have fewer components or perhaps more components than that shown in **Figure 8**.

[0087] At least certain embodiments of the disclosure may be part of a digital media player, such as a portable music and/or video media player, which may include a media processing system to present the media, a storage device to store the media and may further include a radio frequency (RF) transceiver (*e.g.*, an RF transceiver for a cellular telephone) coupled with an antenna system and the media processing system. In certain embodiments, media stored on a remote storage device may be transmitted to the media player through the RF transceiver. The media may be, for example, one or more of music or other audio, still pictures, or motion pictures.

[0088] The portable media player may include a media selection device, such as a click wheel input device on an iPod® or iPod Nano® media player from Apple, Inc. of Cupertino, CA, a touch screen input device, pushbutton device, movable pointing input device or other input device. The media selection device may be used to select the media stored on the storage device and/or the remote storage device. The portable media player may, in at least certain embodiments, include a display device which is coupled to the media processing system to display titles or other indicators of media being selected through the input device and being presented, either through a speaker or earphone(s), or on the display device, or on both display device and a speaker or earphone(s). Examples of a portable media player are described in published U.S. Patent No. 7,345,671 and U.S. Patent No. 7,627,343, both of which are incorporated herein by reference.

[0089] Portions of what was described above may be implemented with logic circuitry such as a dedicated logic circuit or with a microcontroller or other form of processing core that executes program code instructions. Thus processes taught by the discussion above may be performed with program code such as machine-executable instructions that cause a machine that executes these instructions to perform certain functions. In this context, a "machine" may be a machine that converts intermediate form (or "abstract") instructions into processor specific instructions

(*e.g.*, an abstract execution environment such as a "virtual machine" (*e.g.*, a Java Virtual Machine), an interpreter, a Common Language Runtime, a high-level language virtual machine, etc.), and/or, electronic circuitry disposed on a semiconductor chip (*e.g.*, "logic circuitry" implemented with transistors) designed to execute instructions such as a general-purpose processor and/or a special-purpose processor. Processes taught by the discussion above may also be performed by (in the alternative to a machine or in combination with a machine) electronic circuitry designed to perform the processes (or a portion thereof) without the execution of program code.

[0090] The present disclosure also relates to an apparatus for performing the operations described herein. This apparatus may be specially constructed for the required purpose, or it may comprise a general-purpose device selectively activated or reconfigured by a computer program stored in the device. Such a computer program may be stored in a computer readable storage medium, such as, but not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), RAMs, EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a device bus.

[0091] A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (*e.g.*, a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; etc.

[0092] An article of manufacture may be used to store program code. An article of manufacture that stores program code may be embodied as, but is not limited to, one or more memories (*e.g.*, one or more flash memories, random access memories (static, dynamic or other)), optical disks, CD-ROMs, DVD ROMs, EPROMs, EEPROMs, magnetic or optical cards or other type of machine-readable media suitable for storing electronic instructions. Program code may also be downloaded from a remote computer (*e.g.*, a server) to a requesting computer (*e.g.*, a client) by way of data signals embodied in a propagation medium (*e.g.*, via a communication link (*e.g.*, a network connection)).

[0093] The preceding Detailed Descriptions are presented in terms of algorithms and symbolic representations of operations on data bits within a device memory. These algorithmic descriptions and representations are the tools used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable

of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0094] It should be kept in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the above discussion, it is appreciated that throughout the description, discussions utilizing terms such as "receiving," "selecting," "determining," "collecting," "applying," "entering," or the like, refer to the action and processes of a device, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the device 's registers and memories into other data similarly represented as physical quantities within the device memories or registers or other such information storage, transmission or display devices.

[0095] The processes and displays presented herein are not inherently related to any particular device or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct a more specialized apparatus to perform the operations described. The required structure for a variety of these systems will be evident from the description below. In addition, the present disclosure is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the disclosure as described herein.

[0096] The foregoing discussion merely describes some exemplary embodiments of the present disclosure. One skilled in the art will readily recognize from such discussion, the accompanying drawings and the claims that various modifications can be made without departing from the spirit and scope of the disclosure.

CLAIMS

What is claimed is:

1. A non-transitory machine-readable medium having executable instructions to cause one or more processing units to perform a method to manage performance of a data processing system, the method comprising:

collecting historical I/O information for the data processing system;

receiving I/O information of a current I/O operation of the data processing system; and

determining a power state for the data processing system based on the historical I/O information and the I/O information of the current I/O operation.

2. The non-transitory machine-readable medium of claim 1, wherein the method further comprises:

applying the determined power state to a processor of the data processing system.

3. The non-transitory machine-readable medium of claim 1, wherein the I/O information of the current I/O operation comprises a direction of the I/O operation, wherein the direction of the I/O operation is one of read and write.

4. The non-transitory machine-readable medium of claim 3, wherein, in response to the direction of the I/O operation is write, the current I/O information is ignored in the determining of the power state.

5. The non-transitory machine-readable medium of claim 3, wherein, in response to the direction of the I/O operation is read, the determining of the power state comprises selecting a power state of high power consumption level and low latency.

6. The non-transitory machine-readable medium of claim 1, wherein the I/O information of the current I/O operation comprises a size of the I/O operation, wherein the size of the I/O operation comprises an amount of data involved in the I/O operation.

7. The non-transitory machine-readable medium of claim 6, wherein the current I/O operation is a first I/O operation and the power state is a first power state, the method further comprising receiving I/O information of a second I/O operation and determining a second power state based on the historical I/O information and the I/O information of the second I/O operation, wherein the size of the first I/O operation is smaller than a size of the second I/O

operation, wherein the first power state is determined to have higher power consumption level and lower latency than the second power state.

8. The non-transitory machine-readable medium of claim 1, wherein the I/O information of the current I/O operation comprises a quality of service of the current I/O operation.

9. The non-transitory machine-readable medium of claim 8, wherein the current I/O operation is a first I/O operation and the power state is a first power state, the method further comprising receiving I/O information of a second I/O operation and determining a second power state based on the historical I/O information and the I/O information of the second I/O operation, wherein the quality of service of the first I/O operation is higher than a quality of service of the second I/O operation, wherein the first power state is determined to have higher power consumption level and lower latency than the second power state.

10. A computer implemented method to manage performance of a device, the method comprising:

collecting historical I/O information for the device;

receiving I/O information of a current I/O operation of the device; and

determining a power state for the device based on the historical I/O information and the I/O information of the current I/O operation.

11. The method of claim 10 further comprising entering into the determined power state at the device.

12. The method of claim 10, wherein the I/O information of the current I/O operation comprises a size of the current I/O operation, wherein the size of the current I/O operation comprises an amount of data involved in the current I/O operation.

13. The method of claim 10, wherein the I/O information of the current I/O operation comprises a quality of service of the current I/O operation.

14. The method of claim 10, wherein the I/O information of the current I/O operation comprises a media type of the current I/O operation.

15. The method of claim 14, wherein the determining of the power state comprises selecting a power state of higher power consumption level and lower latency in response to the media type of the current I/O operation is solid state drive, and selecting a power state of lower power consumption level and higher latency in response to the media type of the current I/O operation is hard drive.

16. A device for performance management, the device comprising:

a processor;

a memory coupled to the processor though a bus; and

a process executed from the memory by the processor causes the processor to collect historical I/O information, receive current I/O information of an I/O operation, and determine a power state based on the historical I/O information and the current I/O information.

17. The device of claim 16, wherein the historical I/O information comprises a number of I/O operations over a sample period of time.

18. The device of claim 17, wherein the determining of the power state comprises selecting a power state of higher power consumption level and lower latency in response to the number of I/O operations is high, and selecting a power state of lower power consumption level and higher latency in response to the number of I/O operations is low.

19. The device of claim 16, wherein the historical I/O information comprises an inter-arrival time between I/O operations over a sample period of time.

20. The device of claim 19, wherein the determining of the power state comprises selecting a power state of higher power consumption level and lower latency in response to the inter-arrival time between I/O operations is short, and selecting a power state of lower power consumption level and higher latency in response to the inter-arrival time between I/O operations is long. .

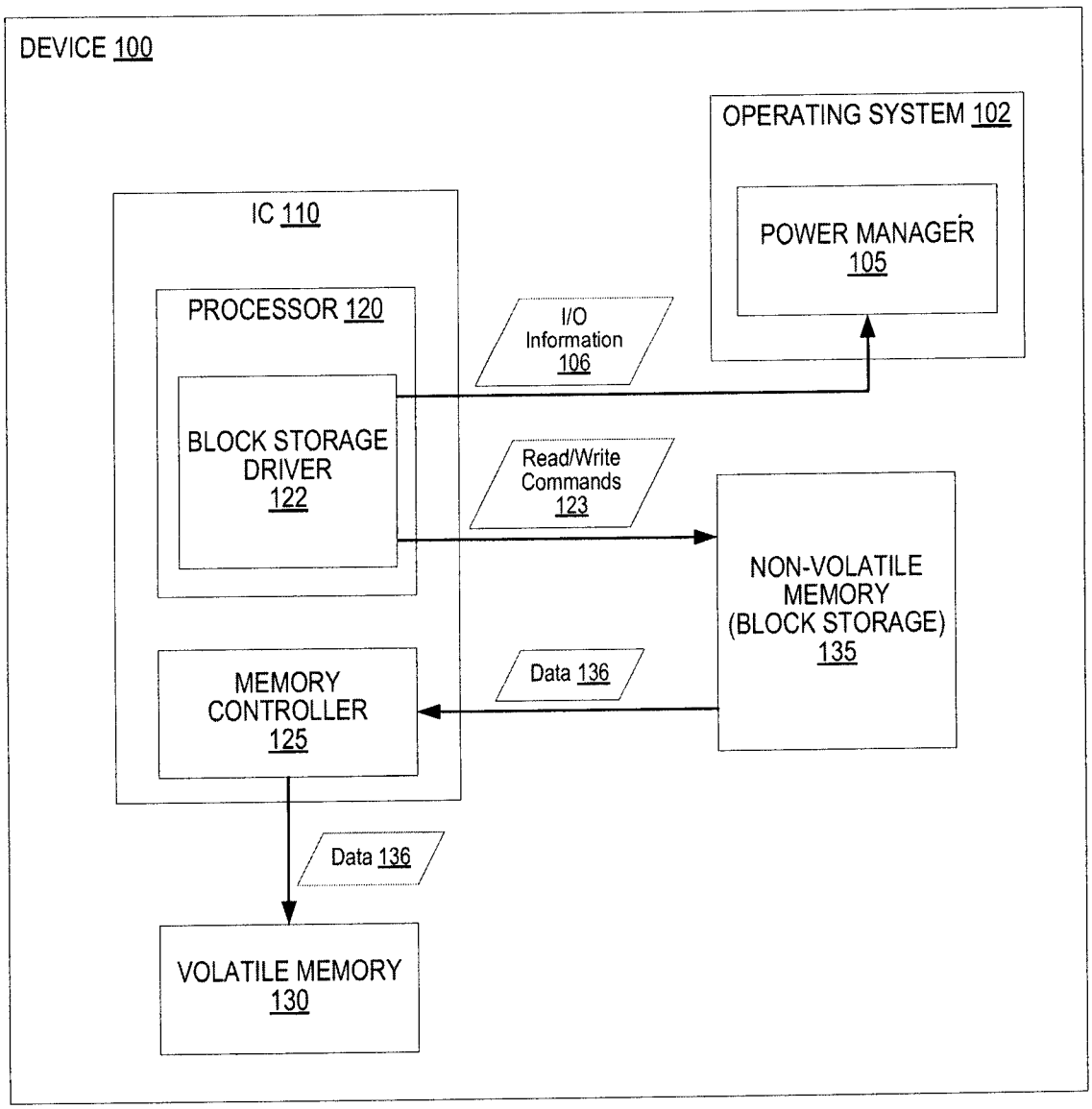
DEVICE 100
OPERATING SYSTEM 102
POWER MANAGER 105
IC 110
PROCESSOR 120
I/O Information 106
BLOCK STORAGE DRIVER 122
Read/Write Commands 123
NON-VOLATILE MEMORY (BLOCK STORAGE) 135
MEMORY CONTROLLER 125
Data 136
Data 136
VOLATILE MEMORY 130

FIG. 1

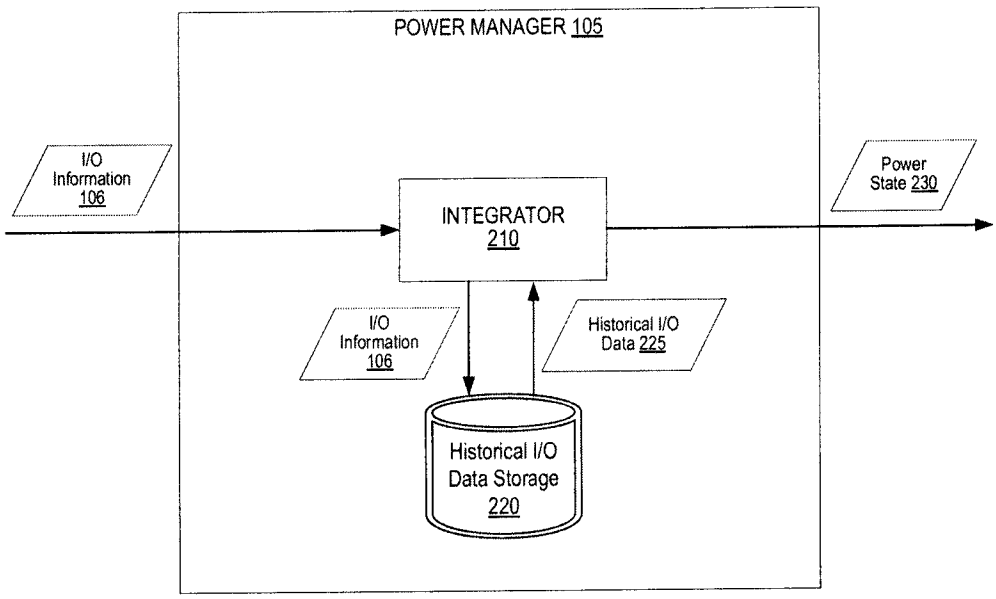
POWER MANAGER 105
I/O Information 106
INTEGRATOR 210
Power State 230
I/O Information 106
Historical I/O Data 225
Historical I/O Data Storage 220

FIG. 2

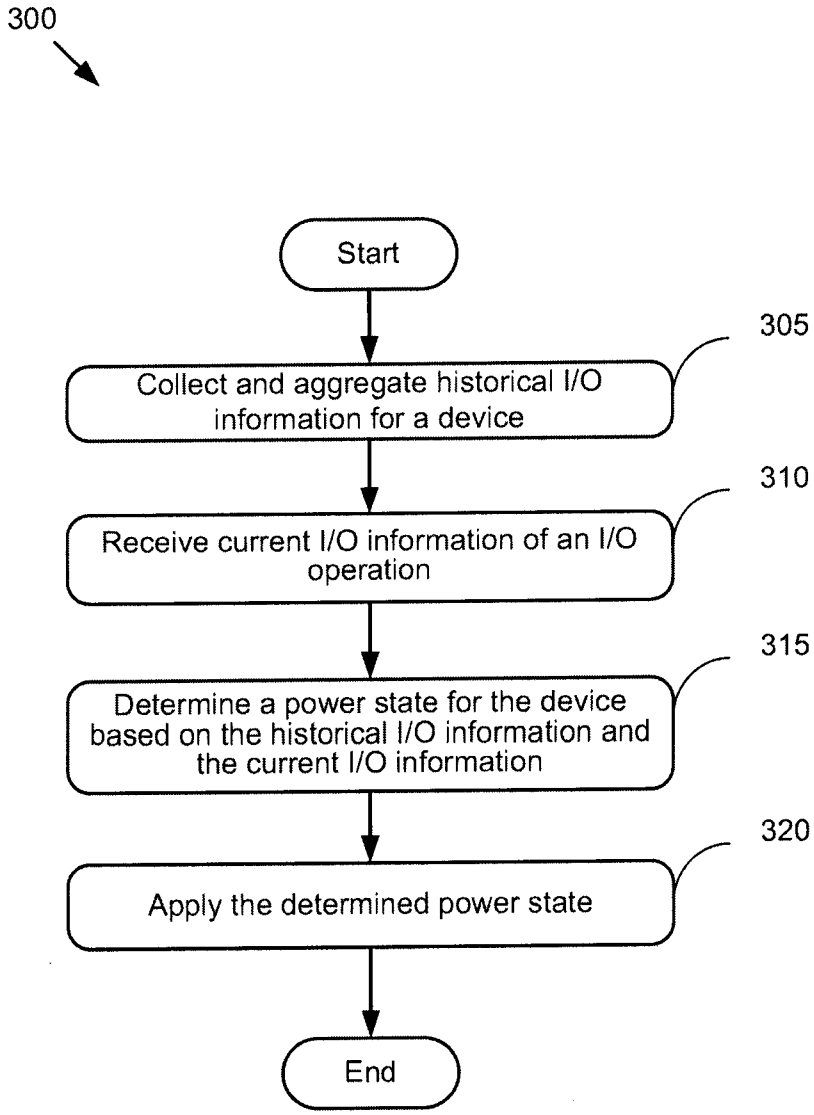
300
Start
Collect and aggregate historical I/O information for a device
305
Receive current I/O information of an I/O operation
310
Determine a power state for the device based on the historical I/O information and the current I/O information
315
Apply the determined power state
320
End

FIG. 3A

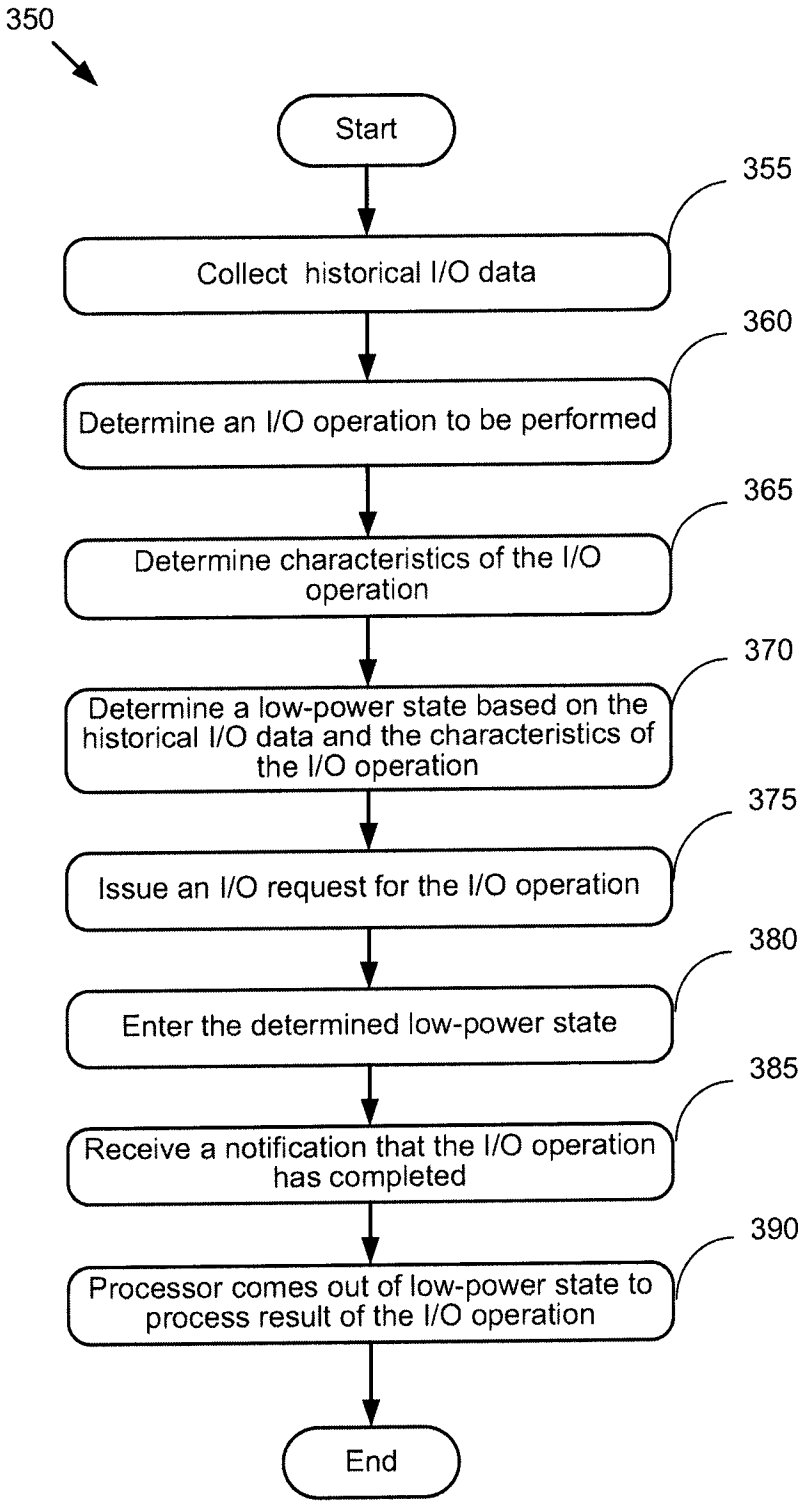
350
Start
355
Collect historical I/O data
360
Determine an I/O operation to be performed
365
Determine characteristics of the I/O operation
370
Determine a low-power state based on the historical I/O data and the characteristics of the I/O operation
375
Issue an I/O request for the I/O operation
380
Enter the determined low-power state
385
Receive a notification that the I/O operation has completed
390
Processor comes out of low-power state to process result of the I/O operation
End

FIG. 3B

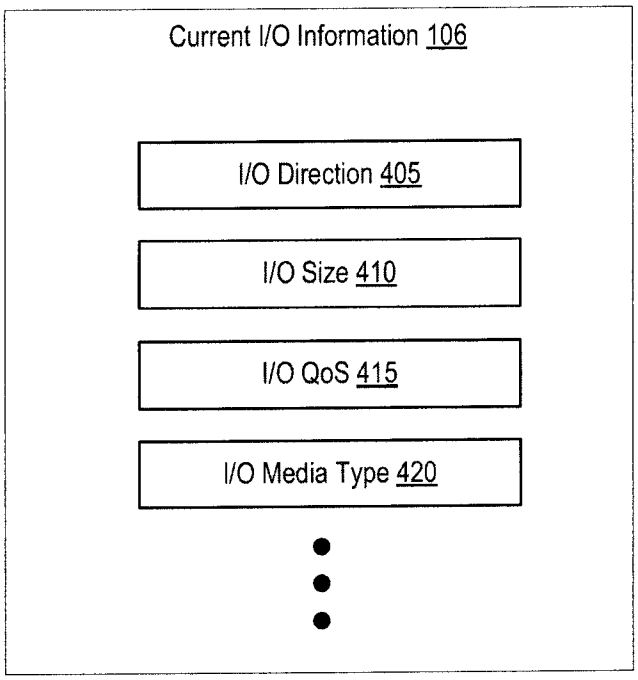
Current I/O Information 106
I/O Direction 405
I/O Size 410
I/O QoS 415
I/O Media Type 420

FIG. 4

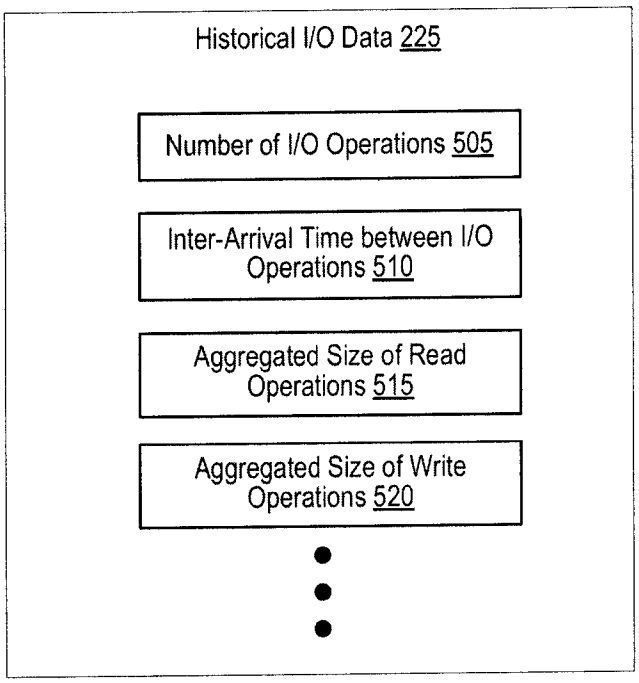
Historical I/O Data 225
Number of I/O Operations 505
Inter-Arrival Time between I/O Operations 510
Aggregated Size of Read Operations 515
Aggregated Size of Write Operations 520

FIG. 5

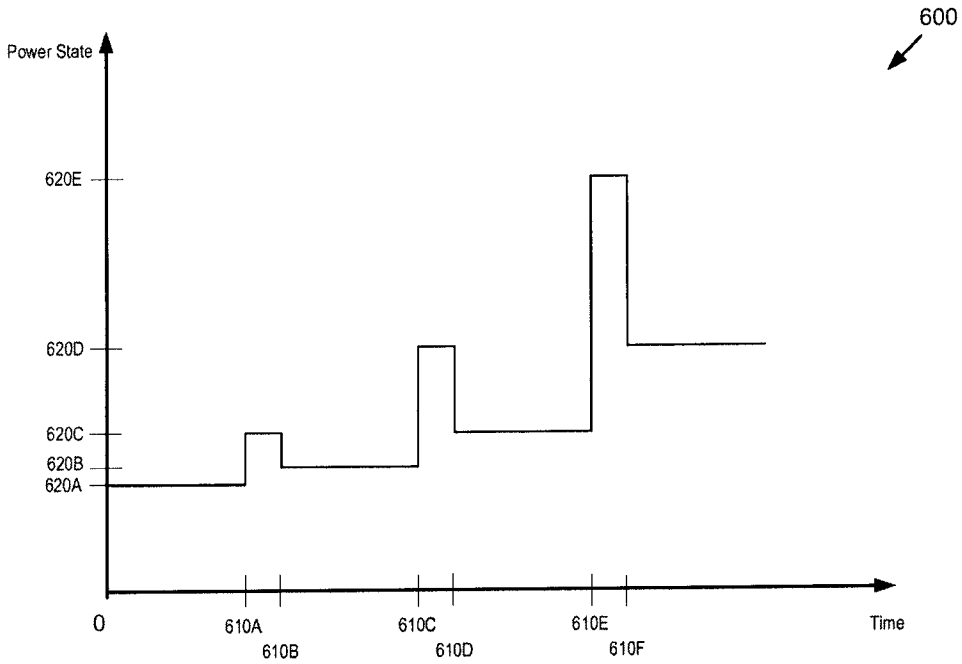
600
Power State
620E
620D
620C
620B
620A
0
610A
610B
610C
610D
610E
610F
Time

FIG. 6

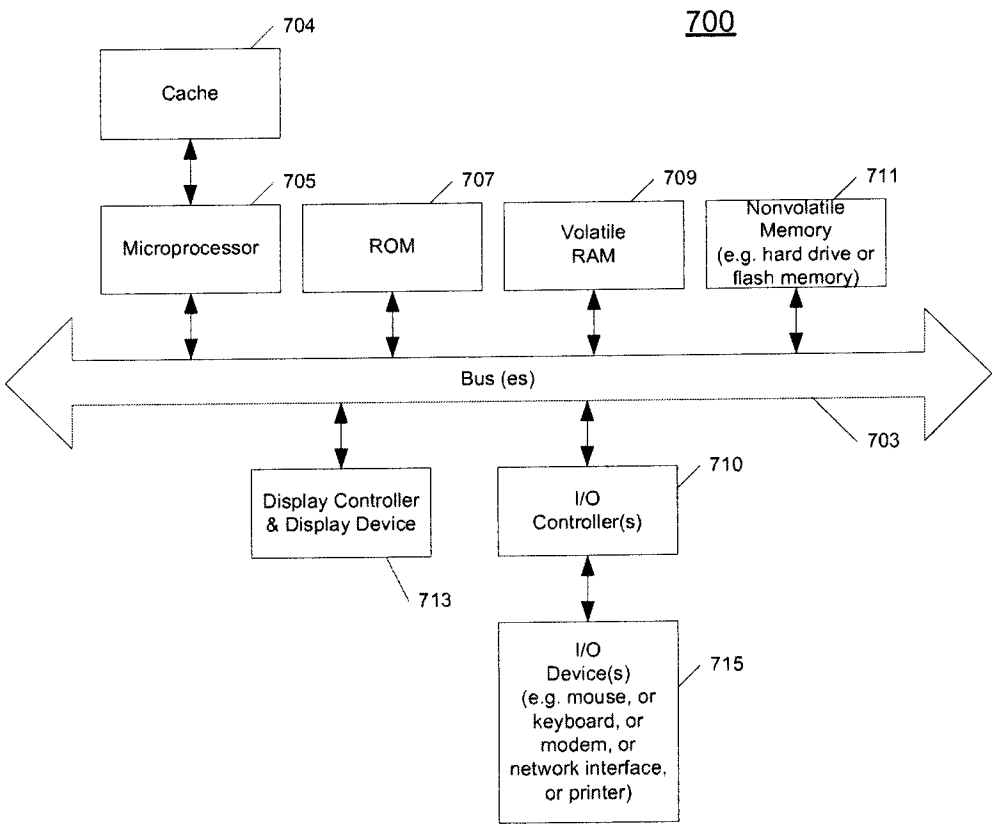
700
704
Cache
705
Microprocessor
707
ROM
709
Volatile RAM
711
Nonvolatile Memory (e.g. hard drive or flash memory)
Bus (es)
703
Display Controller & Display Device
713
I/O Controller(s)
710
I/O Device(s) (e.g. mouse, or keyboard, or modem, or network interface, or printer)
715

FIG. 7

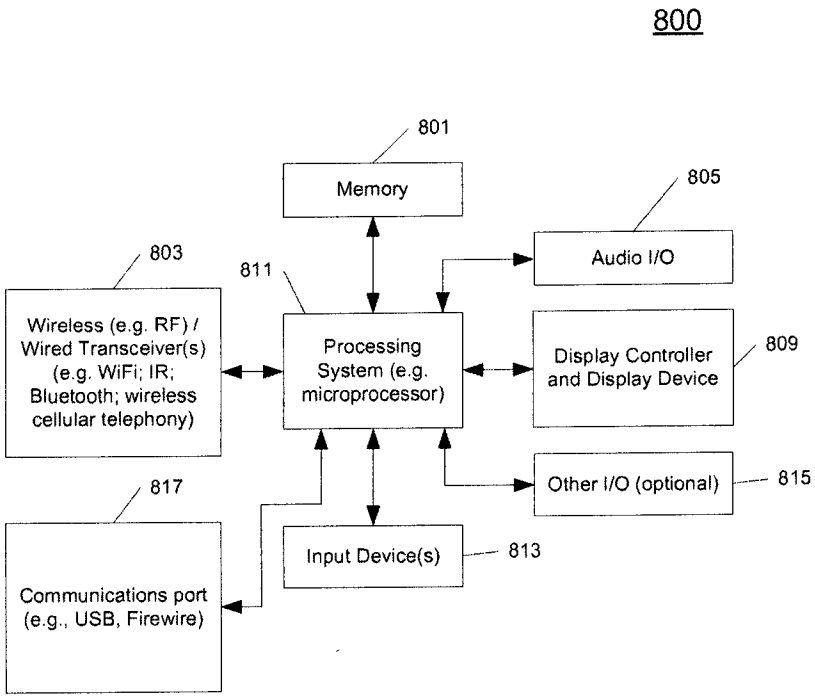
800
801
Memory
805
Audio I/O
803
811
Wireless (e.g. RF) / Wired Transceiver(s) (e.g. WiFi; IR; Bluetooth; wireless cellular telephony)
Processing System (e.g. microprocessor)
Display Controller and Display Device
809
Other I/O (optional)
815
817
Input Device(s)
813
Communications port (e.g., USB, Firewire)

FIG. 8