

# **SYSTEM POWER MANAGEMENT** **USING OSPM/ACPI**

Ashish Deokule



## **SYSTEM POWER MANAGEMENT USING OSPM/ACPI**

Words like Standby, Hibernate, Restart and Shutdown form a pervasive vocabulary in our technical jargon. Has anybody wondered what exactly happens when we click on one of those four tabs? Metaphorically, standby can be visualized as a nap on an antique chair. A person can be up and running in almost no time. Compare this with an afternoon nap. We need to have some afternoon beverage to get us recharged. That's Hibernate for you. And then there's one night's good sleep and we wake up every morning starting with all the routine and fully charged up. Does it sound like Restart or Shutdown? You even have an alarm clock to wake you up just in case anything turns up which require your attention.

These are nothing but different power management policies implemented by a joint venture of hardware and software. It is important to have proper power management policies to utilize a system efficiently. These are not directed towards power preservation only. Its intent is also focused for faster restoration of a working system for the user.

Power Management can be of two types:

- ❖ Incorporated in the device itself. Device driver intervention needed.
- ❖ Incorporated in Operating System. This is called Operating System-directed Power Management(OSPM)

This article will mainly concentrate on OSPM rather than device directed power management. Device directed power management is slowly and gradually becoming obsolete in the system. All leading OS like Windows and Linux are OSPM-centric.

### **Why OSPM?**

OSPM provides some fundamental edge over driver directed power management. Here are few advantages over driver directed power management:

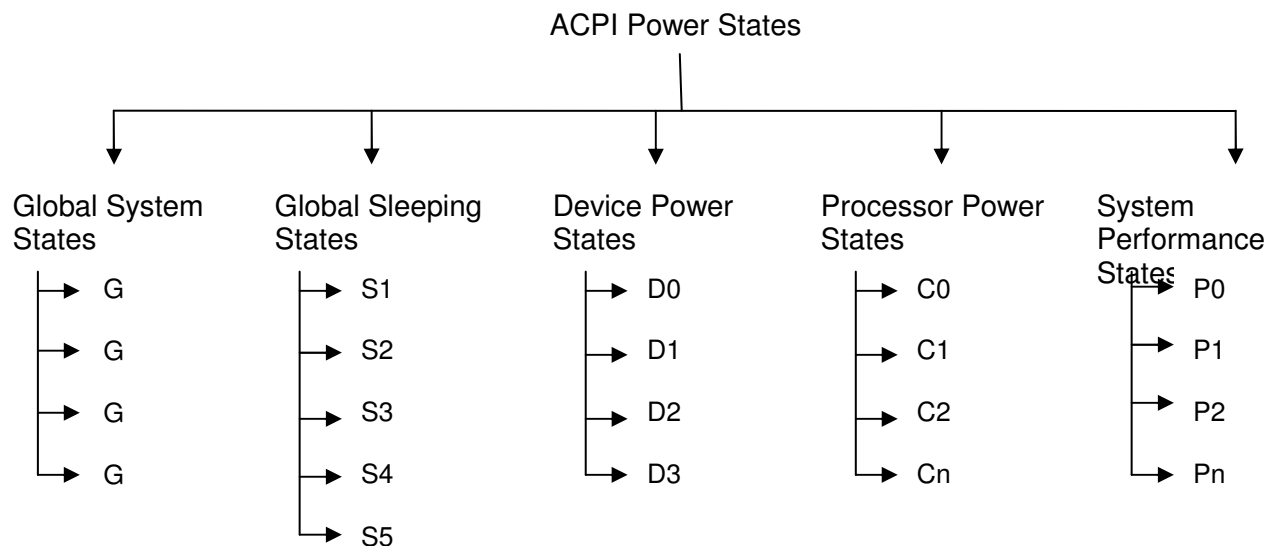
- ❖ Moving power management functionality in the OS itself will help all the platforms and devices to utilize it. Power management capabilities are available to all the systems on which the OS is installed. This can be exploited to the maximum by system and firmware designers.
- ❖ This will reduce the industry wide practice of investing in power management algorithms in the device driver/device itself. As such common algorithms will be incorporated in the OS, organizations can think of innovations rather than simple parity
- ❖ Limitation of implementing of complex power management algorithms in the BIOS has been overcame
- ❖ Development of OSPM can be done independent to development of OS. This abstraction is a key eye-catcher which has interested many major players

Note that OSPM is a concept to incorporate power management functionality in the Operating System. To achieve this, we need to have a standard which is followed by the Operating System and the different device drivers know about it. This void is being filled by a standard developed by behemoths like Intel, Microsoft, HP, Toshiba and Dell. It is called Advanced Configuration and Power Interface, abbreviated as ACPI.

So now it's time to see how Operating system achieves such robust power management. For this we need to get deep into ACPI specification. So fasten your seat belts for a short and brief but eventful ride with ACPI.

To manage power in a better way, ACPI defines many power states for an entire system, device, processor, etc. It then interacts with the User Interface and hardware to decide dynamically which power management policy to be used. Let us take a bird's eye view on the same.

## Power Management States as per ACPI



### Global System States

Global System states are entirely visible to the user. It is principally categorized as per different criteria like applications running, power consumed, various exit latencies after application response and possibility of disassembling the computer.

G0 is completely working state. The system is fully functioning with absolutely no power conservation. The User interface is completely active.

G1 is a sleeping state with the best resume latency. Some rudimentary amount of power is conserved. Here work can be restarted without booting the OS. The system seems to be off from user's perspective. However the applications and backend process are running and doing their job well. The system wakes up after a pre-configured event occurs, which can be from an application interface or a Hardware directly.

G2 is a soft off. It's a computer state where only minimal power is used. This state requires large latency to return back to normalcy. No system context will be present in any way. Here the system must be restarted to come back to working state. Software and GUI intervention is required in this.

G3 is mechanical off. This is a state reached by mechanical means and “not” by software intervention. It is equivalent to unplugging your system abruptly. As a result, the resume latency is extremely large due to rebooting the entire system. Except RTC, there is virtually no current flowing in the entire circuitry.

## **System Sleeping States**

Sleeping states are only applicable in Global G1 state.

S1 is extremely low latency wake state. In this state, no system context is lost. CPU, chipset and cache conditions are preserved.

S2 is again a low wake latency state with wake latency more than S1. Here CPU and cache context is lost. The responsibility of maintaining this context rests with the OS.

S3 is again a low wake latency state where the entire system context is lost except system memory. CPU, cache and chip set context are lost in this state. Hardware maintains only a small portion of CPU context and L2 cache.

S4 is the lowest power consumption state with highest wake latency. This is a special system state which allows system context to be saved and restored gradually. If the system has been commanded to enter S4, the OS will write the entire system context to a file on a non-volatile media like a hard disk with appropriate context markers. After this procedure when you press the ON key of your machine, BIOS will check for any non-volatile sleep data on a NVS file. This will result in restoring the system context provided there is no system configuration change and the user has not manually aborted the restore.

## **Device Power States**

D0 is fully active state. The device is performing in fully functional mode. All the capabilities of that device are active.

D1 is a lower power state with power consumption less than D0 and highest maintenance of system context. D2 is a low power state with power saving more than D1 but lesser context saved than D1.

D3 is a fully off state. The power resources related to that device is off. No device context is maintained in this state.

Note that the exact power savings and context maintenance in D1 and D2 are implementation specific. Furthermore, these power states are applicable only when the Global State G0. In lower Global States, the device states are mostly in D3.

## **Processor Power States**

C0 is the highest power consuming state of a processor. It is the only state in which a processor is executing instructions.

C1 is a lower power state in which processor stops executing instructions. The implementation of C0 and C1 is mandatory as per ACPI specification. However it does give a provision of

developing up to 16 different processor power states; where the differentiation of each lies in the processor itself. Further reduction of power consumption i.e. from C2 to C16 can have no context maintained for any of system cache, CPU or platform.

Note that the processor power states are applicable only in Global state G0.

## **Performance States**

Performance states are various tradeoffs between performance and power consumption while the processor is in C0 state and devices in D0. ACPI allows the system designer to have as many as 16 performance states where P0 is the highest performance highest power consumption state.

I think now it's important that we switch over to the co-relation of this theory to the user interface of the Operating System.

## **Correlation with Graphics User Interface**

It might seem amusing to know that your processor, which is the heart of your system, may stop executing instruction even when the system is fully active. So let us take an example. While reading a PDF and no other application running, what should be the expectation from a processor? Once the required application is loaded in RAM and active, the processor can definitely remain idle. In that case the Operating System just instructs the processor not to execute any instruction. Thus the processor goes in C1 state. This is achieved by executing HLT instruction, which stops the processor from executing an instruction.

Further lower processor states can be entered in different states but that is Operating System specific capability. Other lower states of processor can be reached when the system is really idle from user's point of view i.e. when display is visible but no operation is done by user's intervention. This is of course before the screensaver starts (assuming screensaver scheduling time is fairly high). This idle time is configured and monitored by OSPM using a PM Timer.

Also, do note that all the devices need not be active all the time. Do we require the internal modem or WLAN adapter to be always ON. No. It would only be required when an internet access is being gained through them. As a result, the devices that do not contribute to system performance or use are kept in lower power states like D3. Other example can be of CD ROM Drive. It always is in D3 state. As soon as we click on Eject or press the button on its slider, it sends a wake signal to the OS which brings the device back to D0 and then the slide opens. Note that D states are independent of global states.

Similar cases can be seen for audio and video codec chips. When any media application like Winamp or Windows Media Player is executed these devices get activated by a wake mechanism and then process audio or video data.

No sooner the screensaver appears than the system is said to be in sleeping state i.e. G1. However the sleeping state is S1. Here from OS's perspective all the applications are running, but from as far the user is concerned, the OS and system is idle. Here, the wake is usually generated by a mouse movement or keyboard. No data loss happens as is evident.

Other sleeping states like S2/S3 would be reached by logging out (i.e. Log Off) or in Standby mode. When the device is in standby mode or the screen is locked, OS directs the system to go in S3 state. As only system memory context is saved, all the applications running and files open will remain in the current state.

When the system is directed to go in S4, one will have to click on Hibernate. It saves entire working environment on a non-volatile storage media like a hard disk. And hence while rebooting the system you get all the files in the same condition i.e. in unsaved, open fashion without even rebooting the system. Obviously, the resume time is much longer.

G2 Soft Off or S5 system state is reached by software initiated shutdown. When the system is shut down directly by removal of power by unplugging, the system state is said to be G3. The difference between G2 and G3 is one is gradual and complete shutdown whereas the other is abrupt and incomplete.

To summarize, OSPM is extremely efficient concept to save the investment done by different organizations on power management algorithms. As described above, the power consumption and conservation policies can be put to a perfect execution if and only if different devices just do what the OS wants from the System perspective. Further, all the action regarding maintenance of power from user's aspect as mentioned above, happening deep inside a system should make a user utilize the system in a better way.