Power Management In The Linux* Kernel

Current Status And Future

Rafael J. Wysocki

Intel Open Source Technology Center

September 18, 2013

Outline

What We Are After

Goal (Holy Grail of Computer PM)

Use only as much energy as needed to achieve sufficient performance.

What software can do

(1) Figure out how much performance/capacity is needed and how much latency is acceptable, (2) deliver what's necessary and turn off everything else (if possible).

What about the Linux kernel?

The Kernel's Role

Provide the ability to turn things off

- The "mechanics" (carrying out transitions).
- Signaling events (e.g. wakeup).
- Cost information.

Estimate what's needed

- Follow the actions of user space.
- Use information available internally (e.g. from the CPU scheduler).
- Follow trends.

System-Wide PM and Runtime PM

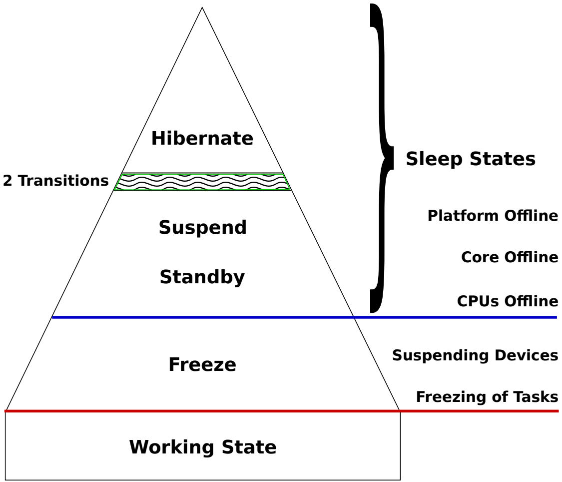
System-wide power management

- Global energy-saving states.
- Whole system PM transitions.
- User space decides which energy-saving state to go to and when.

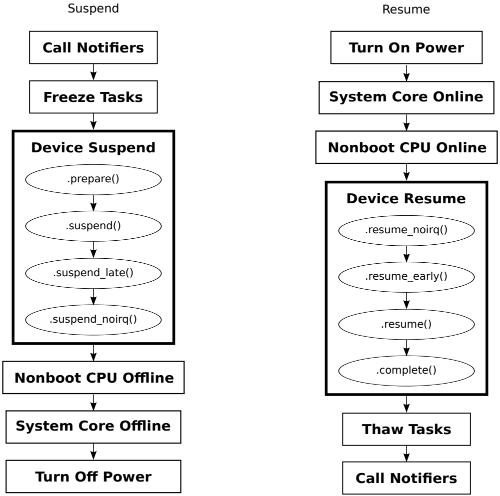
Runtime (working state) power management

- User space processes run.
- Individual CPU or I/O device energy saving utilized.

System-Wide PM Overview

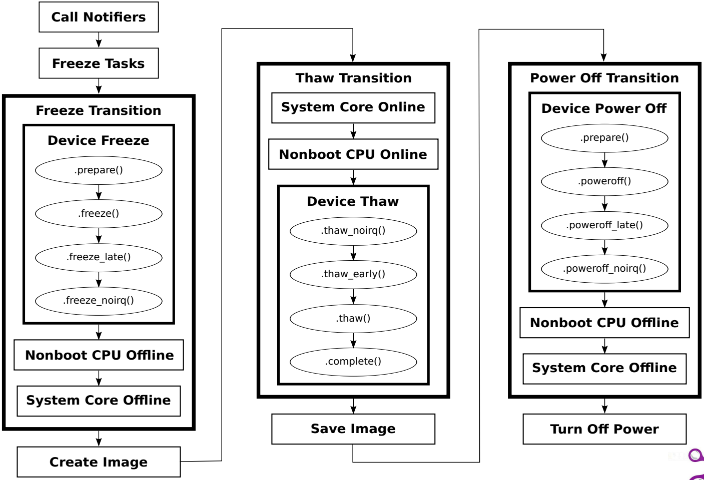


System Suspend/Resume Control Flow

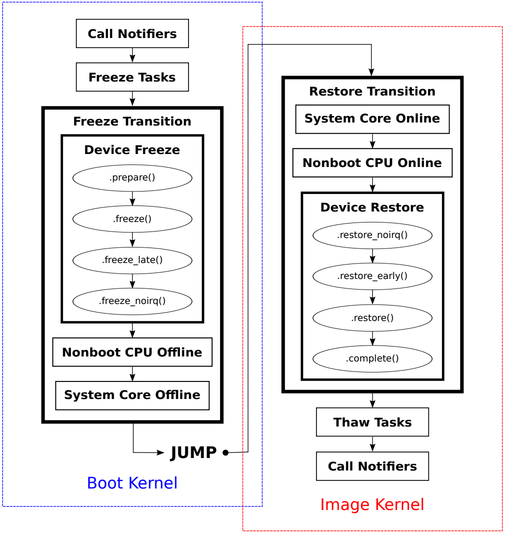


INTEL OPEN SOURCE TECHNOLOGY CENTER

System Hibernation Control Flow



System Restore Control Flow



System-Wide PM Summary

Global energy-saving states, “frozen” user space

Freeze, Standby, Suspend, Hibernate.

Controlled by user space

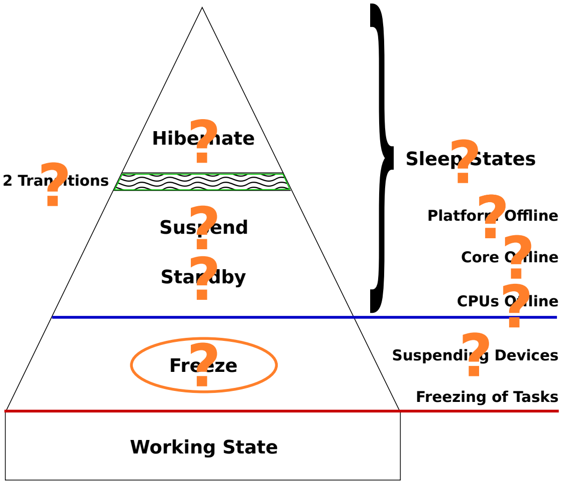
1. User space selects the target state.
2. User space decides when to start transitions.
   - Direct command (*sysfs* write).
   - “Wakelocks” interface.

Used on many systems

Dektop distributions, Android (system suspend + “wakelocks”).

INTEL OPEN SOURCE TECHNOLOGY CENTER

The Future Of System-Wide PM



The Future Of System-Wide PM – Summary

Hibernation's future uncertain

- Replaced by technologies like Intel Rapid Start.
- Not implemented on ARM.

Future platforms may not support "platform offline"

- Suspend and standby may not make sense (no sleep states).
- CPUs offline may not be necessary.
- Freeze may be used to leverage the existing infrastructure.

Freeze may not be useful if runtime PM is very efficient

INTEL OPEN SOURCE TECHNOLOGY CENTER

Working-State (Runtime) PM Frameworks – CPU

CPUidle: CPU C-states framework

Idle CPU cores go into low-power states automatically (the low-power state to go to depends on when the CPU core is going to be needed again and on wakeup latency requirements).

CPU performance scaling

Automatic switching performance levels of CPU cores depending on load.

- "Traditional" *cpufreq* (on *x86* it uses ACPI P-states).
- `intel_pstate` (governor and scaling driver combined).

CPU Power Management Problem

CPU performance scaling and CPUidle are separate

CPUidle: Select the most suitable (low-power) state.

CPUfreq: Balance CPU load and performance level.

However, they are not independent

- CPUidle depends on the CPU scheduler.
- CPU performance scaling affects the CPU scheduler.

Question: What is more efficient?

- Doing work in bursts and going idle between them.
- Doing work at a suitable performance level without going idle.

The Future Of CPU Power Management

Multiple CPU packages add complexity

Many different scheduling strategies are potentially viable.

PM-Aware Scheduling Conjecture

Energy efficiency may be improved without hurting performance by making the CPU scheduler take active role in CPU power management.

Predictions (usual disclaimers apply)

- PM-aware scheduling will be investigated and possibly implemented.
- CPU performance scaling and CPUidle will be combined.

INTEL OPEN SOURCE TECHNOLOGY CENTER

Working-State (Runtime) PM Frameworks – I/O Devices

Likely to become more and more important over time

System-wide PM may be less useful on future systems.

Device Runtime PM API

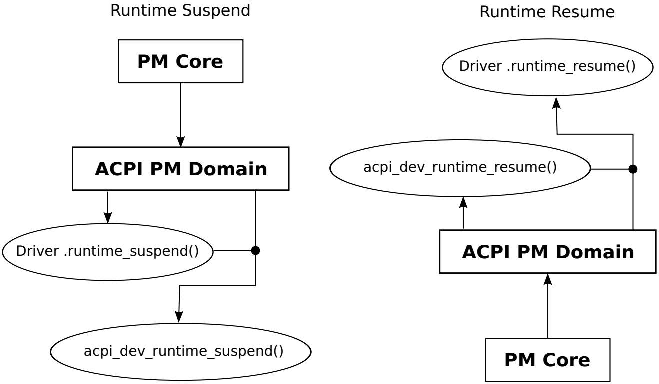
Unified API allowing device low-power states to be used consistently involving device drivers, subsystems and the platform.

devfreq (I/O Device Frequency Scaling)

Automatic voltage and frequency scaling for I/O devices depending on performance requirements (uses the Operation Performance Points API).

INTEL OPEN SOURCE TECHNOLOGY CENTER

Example: Platform Device Runtime PM (ACPI)



Runtime PM On Future Platforms

Hardware design trends

- Increasing integration of components.
- Increasing level of support for aggressive energy saving.

System-on-a-Chip (SoC) configurations

- CPU packages containing I/O devices (e.g. Intel Haswell ULT).
- I/O device states affect package C-states.
- PM features of different components are interdependent.

Challenge: To define clean interfaces for user space.

INTEL OPEN SOURCE
TECHNOLOGY CENTER

Conclusion

- The Linux kernel supports power management in a number of ways.
- Both system-wide and working state (runtime) PM are supported.
- Support for system-wide PM in device drivers is generally better.
- I/O device runtime PM support improving, but there are issues.
- CPU PM is well supported, more integration possible.
- The (long-term) future of system-wide PM is uncertain.
- Hardware design trends increase PM complexity.
- Interdependencies between PM features are challenging.

Questions?

INTEL OPEN SOURCE TECHNOLOGY CENTER

References

- J. Corbet, *The cpuidle subsystem* (`http://lwn.net/Articles/384146/`).
- R. J. Wysocki, *Why We Need More Device Power Management Callbacks* (`https://events.linuxfoundation.org/images/stories/pdf/lfcs2012_wysocki.pdf`).

INTEL OPEN SOURCE
TECHNOLOGY CENTER

Documentation And Source Code

- Documentation/cpu-freq/*
- Documentation/cpuidle/*
- Documentation/power/devices.txt
- Documentation/power/pci.txt
- Documentation/power/runtime_pm.txt
- include/linux/cpufreq.h
- include/linux/cpuidle.h
- include/linux/device.h
- include/linux/pm.h
- include/linux/pm_runtime.h
- include/linux/suspend.h
- drivers/acpi/processor_idle.c
- drivers/base/power/*
- drivers/cpufreq/*
- drivers/cpuidle/*
- kernel/power/*

Legal Information

INTEL OPEN SOURCE TECHNOLOGY CENTER

Thanks!

Thank you for attention!