Integration of PM-runtime with System-wide Power Management

Rafael J. Wysocki

Intel Linux Systems Engineering

September 10, 2019
It Is All About Energy-efficiency
Two Different Ways To Get There
High-level View to PM-runtime

- **Active**
- **Suspended**
- **Resume**
- **Suspend**
- **Is Idle?**

Trigger Event:
- NO

Rafael J. Wysocki
PM-runtime and System-wide PM
September 10, 2019
System Suspend Control Flows

Working State

Call Notifiers
Freeze Tasks
Device Suspend
Device Resume
Nonboot CPU Offline
Nonboot CPU Online
System Core Offline
System Core Online
Platform Offline
Platform Online

Suspend to Idle

Wait For a Wakeup Event

Platform-based Suspend

Rafael J. Wysocki
PM-runtime and System-wide PM
September 10, 2019
Transparent Handling of Suspended Devices
PM-runtime Callbacks Re-Use
Driver Core and Power Management

- **PM Domain**
- **Bus**
- **Middle Layer**
- **Device Driver**
- **Device**
- **Action**
- **Driver Core**
- **Class**
- **Type**

Rafael J. Wysocki
PM-runtime and System-wide PM
September 10, 2019 8 / 24
Devices May Need To Be Reconfigured
Differences Related To Wakeup

PM-runtime
Device wakeup always enabled.

System-wide PM
/sys/devices/.../power/wakeup: enabled or disabled.
Intermittent Interactions Between Devices
Direct-complete Optimization Idea

Working State
- Call Notifiers
- Freeze Tasks
- Device Suspend
  - .prepare()
  - .suspend()
  - .suspend_late()
  - .suspend_noirq()
- Wait For a Wakeup Interrupt
- Device Resume
  - .complete()
  - .resume()
  - .resume_early()
  - .resume_noirq()
- Thaw Tasks
- Call Notifiers

Working State > 0
Limitations Of Direct-complete

1. Devices subject to intermittent interactions cannot do it.
2. Cannot be done if the children do not do it too.
Wrappers Around PM-runtime Callbacks
Limitations Of Callback Wrappers

1. Invoke middle-layer PM-runtime callbacks.
2. Disable PM-runtime.
3. Questionable approach to leaving devices in suspend.
Driver Flags For System-wide Power Management

- DPM_FLAG_NEVER_SKIP
- DPM_FLAG_SMART_PREPARE
- DPM_FLAG_SMART_SUSPEND
- DPM_FLAG_LEAVE_SUSPENDED
Coverage Gaps

1. Runtime resume resulting from intermittent interactions not covered.

2. DPM_FLAG_SMART_SUSPEND (generally) required for callback re-use.
Observation 1
Devices cannot be runtime-suspended during system-wide suspend/resume.

Observation 2
Two cases for runtime resume during system-wide PM transitions:
- Called from system-wide PM callbacks for the same device.
- Called from somewhere else.
If called from system-wide PM callbacks for the same device, run it.

Otherwise:

1. If called after some system-wide PM code has run, block it.
2. Else (always) defer suspending the target device to the “noirq” phase.
Generally, Callback Wrappers Do Not Help
Plan: Eliminate The Reason For Using Callback Wrappers

Concern addressed by them
Prevent the same callback from running twice in a row back to back.

There are cases in which they are not necessary already
- PM-aware bus types (PCI, USB).
- ACPI PM domain.

What can be done elsewhere
1. Make all PM-aware middle-layer code honor PM-runtime status.
2. For pass-through middle layers, make the PM core do that.
Comments, Questions, Concerns?
References

- Rafael J. Wysocki, *PM Infrastructure in the Linux Kernel – Current Status and Future* (https://events.linuxfoundation.org/sites/events/files/slides/kernel_PM_infra_0.pdf).
Disclaimer

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

© Intel Corporation