SSD Auto-Runtime Power Management

Lin Ming <ming.m.lin@intel.com>

June 6, 2012

Demo: disk auto runtime PM

- 2 disks
  sdb: OS disk
  sda: data disk. Demo disk auto runtime PM feature on data disk.

- Prepare
  # enable disk runtime PM feature
echo auto >
/sys/devices/pci0000:00/0000:00:1f.2/ata1/host0/target0:0:0/0:0:0:0/power
/control

  # Set auto suspend delay time, for example, 5 seconds
echo 5000 >
/sys/devices/pci0000:00/0000:00:1f.2/ata1/host0/target0:0:0/0:0:0:0/power
/autosuspend_delay_ms

- Demo steps
  - sda idle for 5 seconds: suspended
  - mount /dev/sda1 /mnt: resumed immediately
  - sda idle for 5 seconds again: suspended
  - copy file to /mnt: resumed immediately

System PM vs Runtime PM

System PM

- system wide suspend/resume
- all devices together
- Initiated by userspace: echo mem > /sys/power/state

Runtime PM

- system is still running
- per device suspend/resume
- controlled by driver

Runtime PM callbacks

```
struct dev_pm_ops {
        ...
        int (*runtime_suspend)(struct device *dev);
        int (*runtime_resume)(struct device *dev);
        int (*runtime_idle)(struct device *dev);
};
```

- Implemented by driver
- Executed by PM core

Runtime PM API

- pm_runtime_suspend(dev), pm_schedule_suspend(dev, delay)
  ->runtime_suspend(dev)

- pm_runtime_resume(dev), pm_request_resume(dev)
  ->runtime_resume(dev)

- pm_runtime_idle(dev), pm_request_idle(dev)
  ->runtime_idle(dev)

- Many other APIs ….

Runtime auto suspend

Suspend device after some period of idle time

- /sys/devices/.../power/autosuspend_delay_ms

Idle is determined based on dev->power.last_busy

- pm_runtime_mark_last_busy() to update

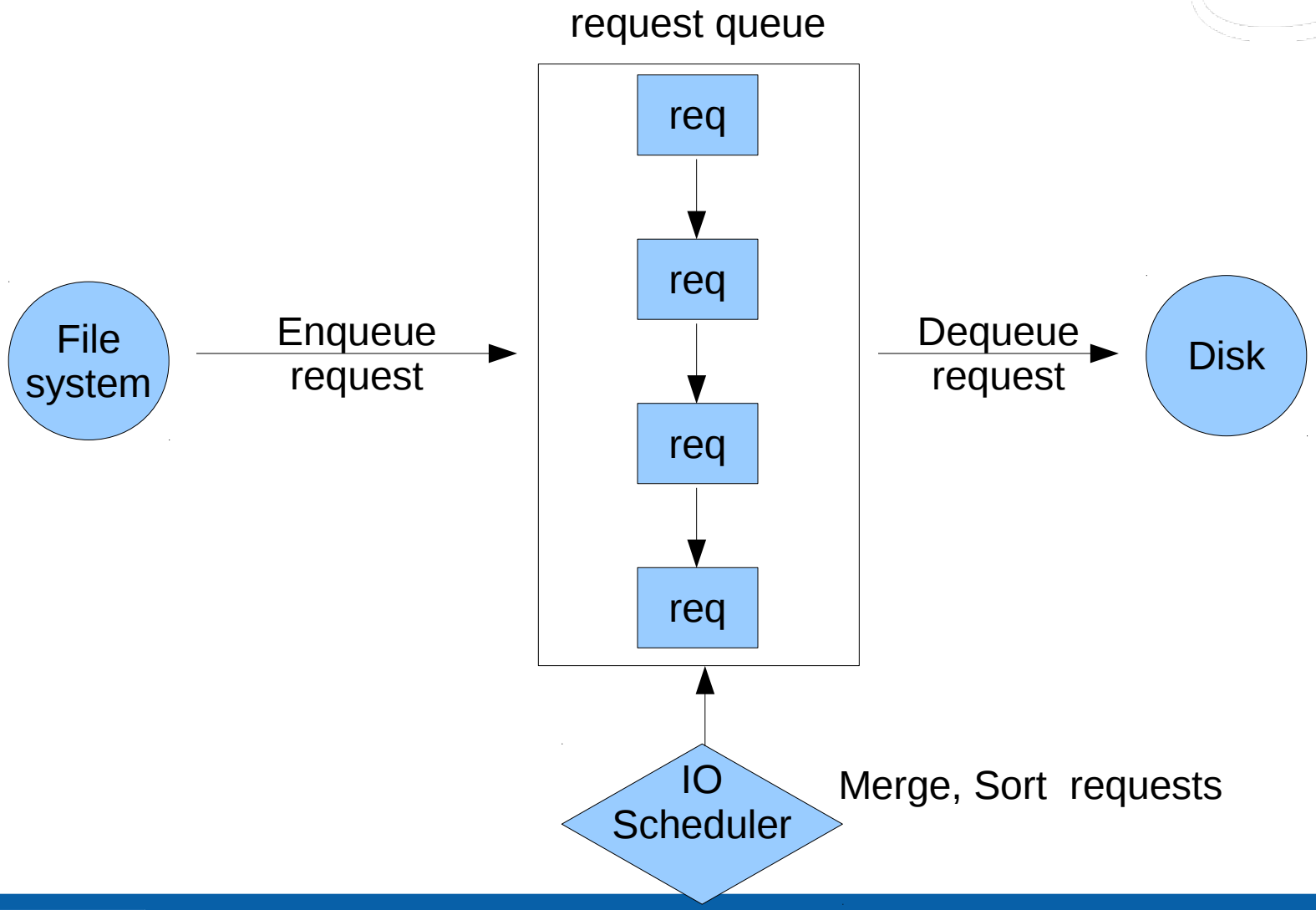
suspend timer

Runtime PM sysfs interface

/sys/devices/pci0000:00/0000:00:1f.2/ata3/host2/target2:0:0/2:0:0:0/power

- async
- autosuspend_delay_ms
- control
- runtime_active_kids
- runtime_active_time
- runtime_enabled
- runtime_status
- runtime_suspended_time
- runtime_usage

Block layer request queue



Block layer runtime PM

Idea came from Alan Stern and Jens Axboe

- "Runtime PM and the block layer", August 2010
- http://marc.info/?t=128259108400001&r=1&w=2

Idea is simple

- Suspend disk when request queue is empty for some time
- When a new request coming, delay handling it until device is resumed

New fields in struct request_queue

- nr_pending: track if queue is empty
- rpm_status: runtime power status
- dev: underlying disk

PM request

- REQ_PM flag
- special request to suspend/resume disk
- always inserted at the head of request queue

Block layer runtime PM API

blk_pm_runtime_init()

- set q->dev
- call pm_runtime_mark_last_busy(), pm_runtime_use_autosuspend(), and pm_runtime_autosuspend().

blk_pre_runtime_suspend()

- If any requests are in the queue, return -EBUSY.
- Otherwise set q->rpm_status to RPM_SUSPENDING

blk_post_runtime_suspend()

- If the suspend succeeded then set q->rpm_status to RPM_SUSPENDED
- Otherwise set it to RPM_ACTIVE and call pm_runtime_mark_last_busy().

Block layer runtime PM API cont.

block_pre_runtime_resume()

- Set q->rpm_status to RPM_RESUMING.

block_post_runtime_resume()

If the resume succeeded then set q->rpm_status to RPM_ACTIVE and call pm_runtime_mark_last_busy() and pm_runtime_request_autosuspend().
Otherwise set q->rpm_status to RPM_SUSPENDED.

API usage example: SCSI sd driver

```
scsi_sysfs_add_sdev
        blk_pm_runtime_init(rq, &sdev->sdev_gendev);

scsi_runtime_suspend
        blk_pre_runtime_suspend(q)

        /* suspend scsi device */

        blk_post_runtime_suspend(q, err)

scsi_runtime_resume
        blk_pre_runtime_resume(q);

        /* resume scsi device */

        blk_post_runtime_resume(q, err);

scsi_runtime_idle
        pm_runtime_mark_last_busy(dev);
        pm_runtime_autosuspend(dev);
```

Request added/peek/finished

When a request is added:

If q->rpm_status is RPM_SUSPENDED, or if q->rpm_status is RPM_SUSPENDING and the REQ_PM flag isn't set, call pm_request_resume().

When pick a request:

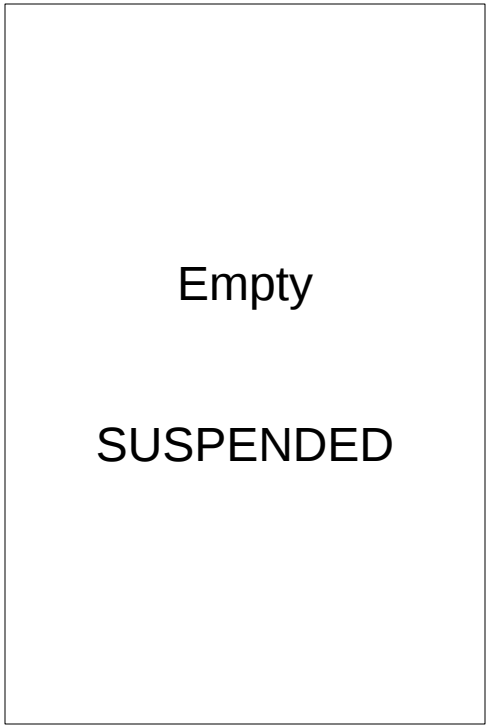
If q->rpm_status is RPM_SUSPENDED, act as though the queue is empty. If q->rpm_status is RPM_SUSPENDING or RPM_RESUMING, only REQ_PM request is sent to driver.

When a request finishes:

Call pm_runtime_mark_last_busy()

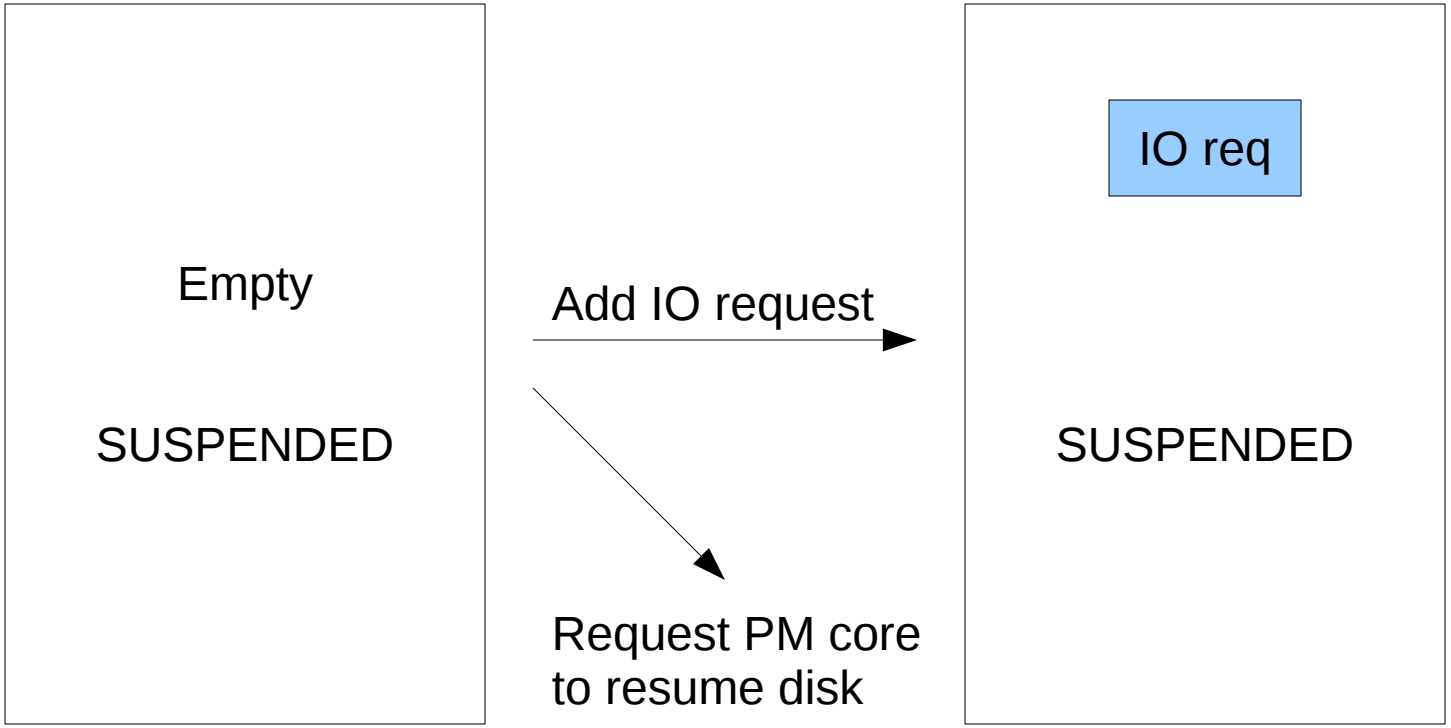
A simple scenario: Initial state

Request queue is empty and suspended



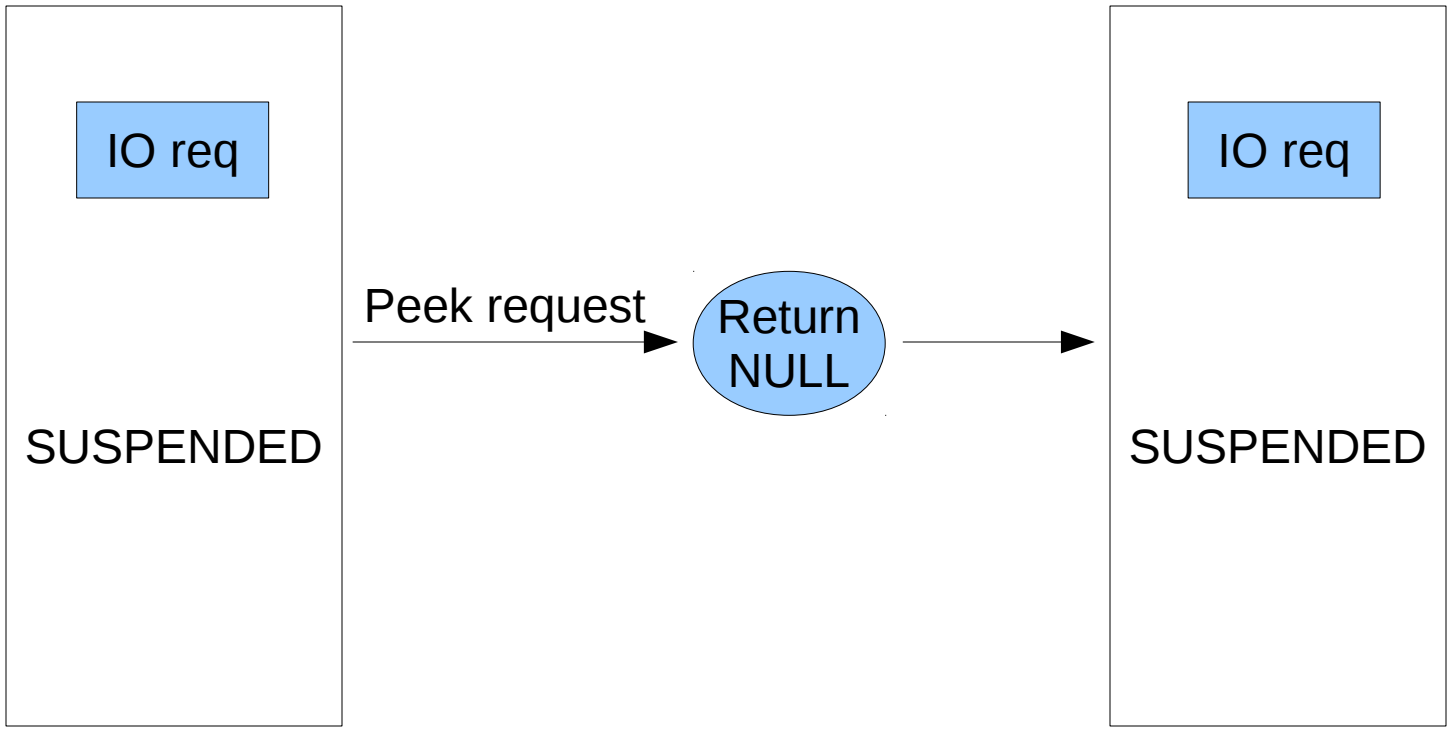
IO request coming

An IO request is coming



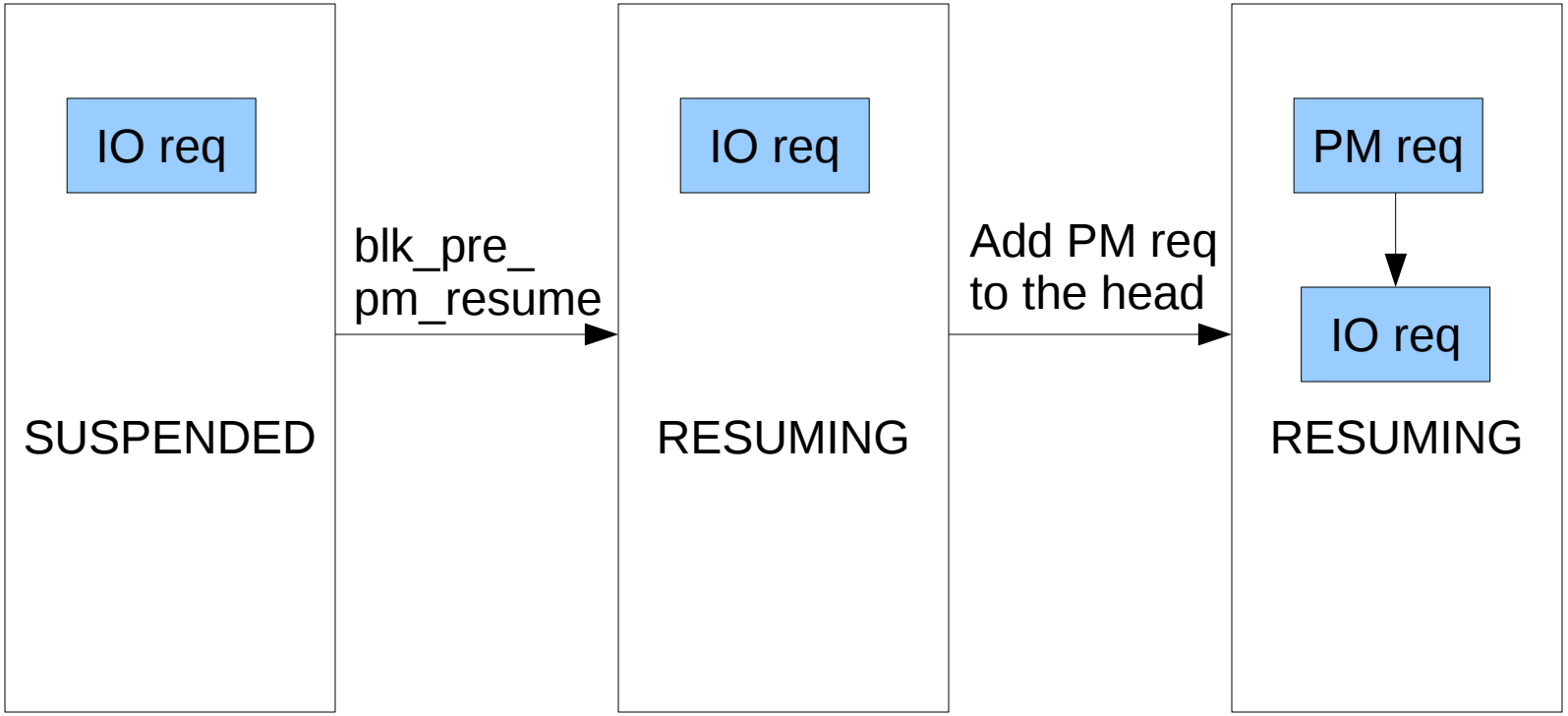
Peek IO request

Block layer try to send IO request to driver



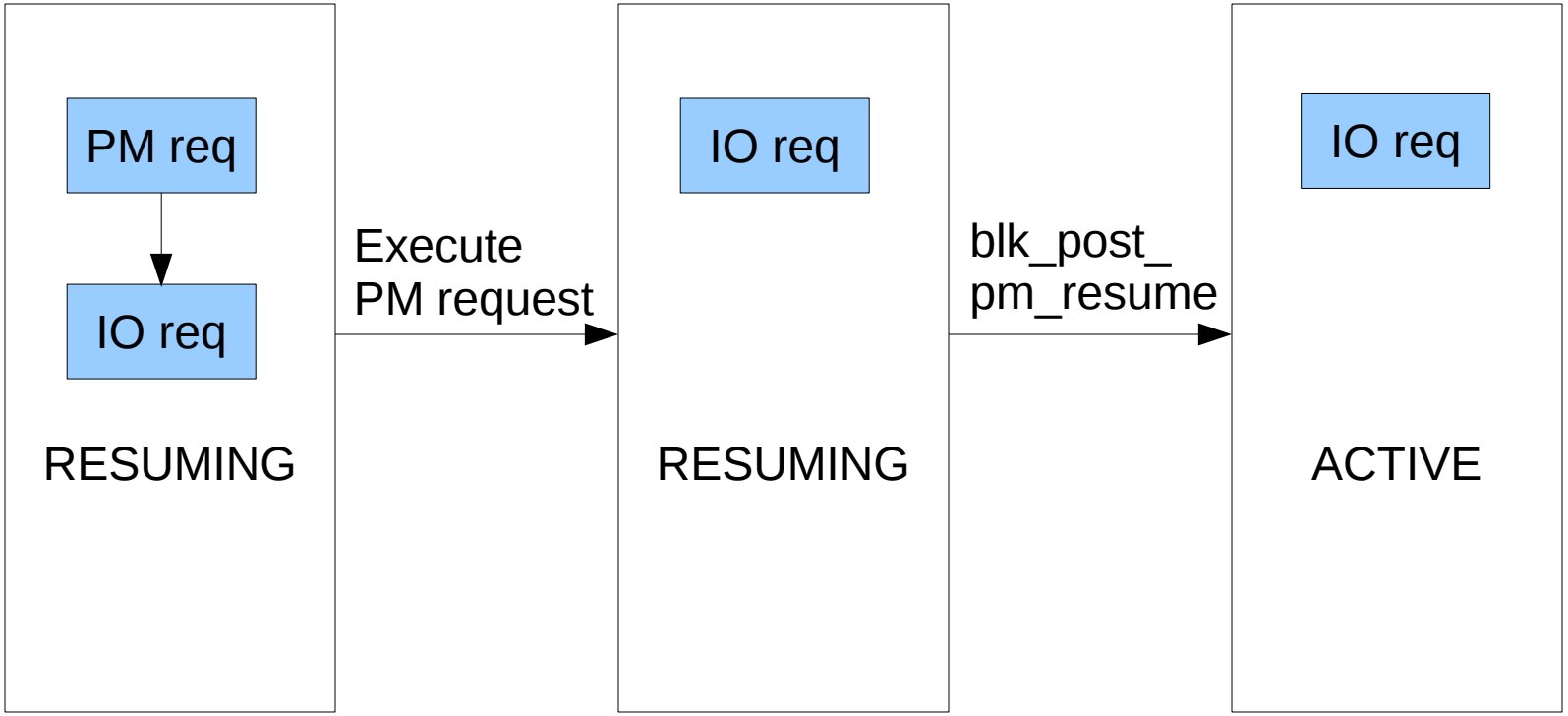
PM request coming

PM req is inserted to resume disk



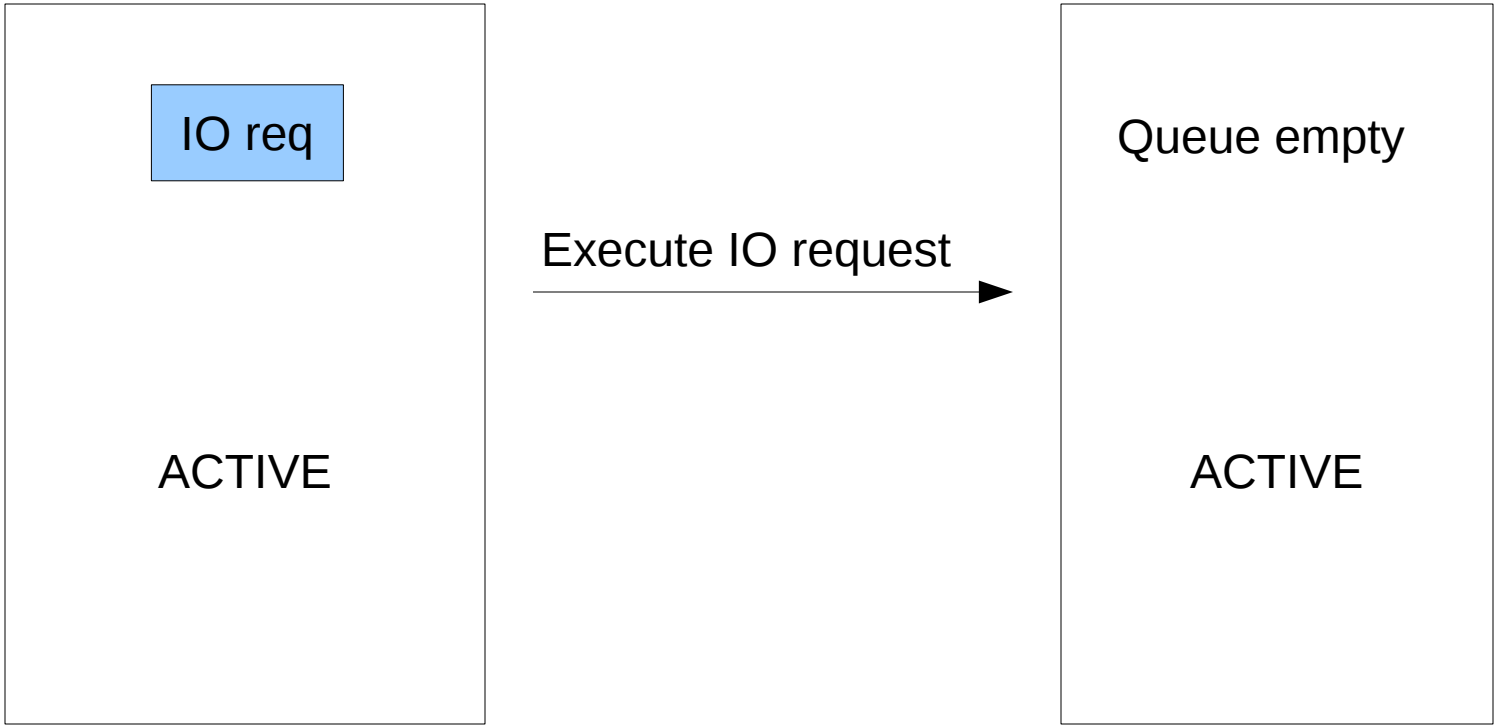
Execute PM request

PM request is sent to resume disk



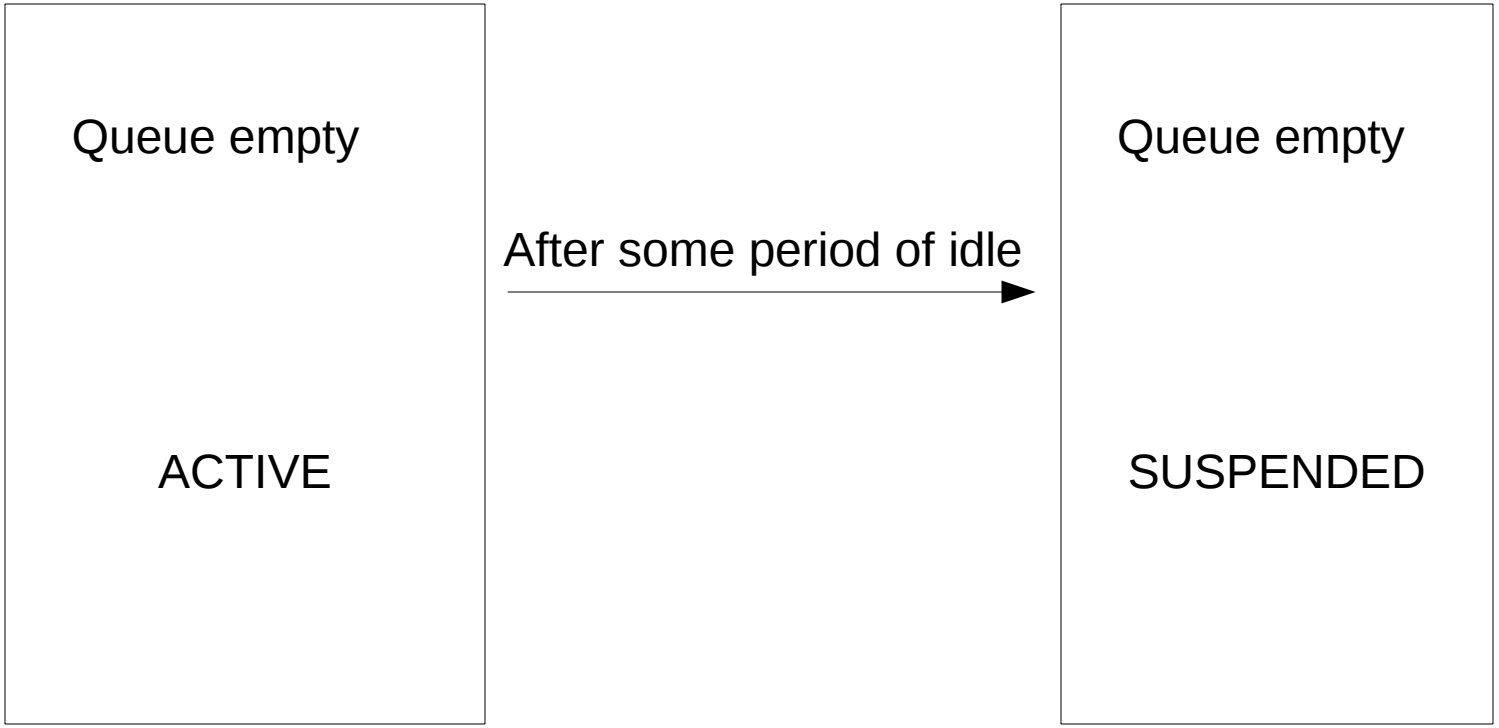
Execute IO request

Disk is active now, IO request can be executed



Disk suspended again

Queue is emtpy now



SSD runtime power off

SSD runtime power off via ACPI

ACPI power resource

- _ON: turn on power
- _OFF: turn off power

```
Method (_ON)
{
    And (GL04, 0xF7, GL04)
    Sleep (0x0A)
}
```

```
Method (_OFF)
{
    Or (GL04, 0x08, GL04)
 }
```

Implemented in “[PATCH v4 06/13] libata-acpi: add ata port runtime D3Cold support”
http://lkml.org/lkml/2012/5/28/13

An issue: printk

Disk resumed by printk:

- scsi_runtime_suspend
- sd_suspend
- printk(....)

  sd 0:0:0:0: [sda] Synchronizing SCSI cache
  sd 0:0:0:0: [sda] Stopping disk
- rsyslogd waken up
- write kernel log to disk
- disk resumed

Ideas? A power friendly printk?

Resource

Source

git pull git://git.kernel.org/pub/scm/linux/kernel/git/mlin/linux.git block_pm

Runtime PM and the block Layer

http://marc.info/?t=128259108400001&r=1&w=2

linux/Documentation/power/runtime_pm.txt

Thanks

Many thanks to Alan Stern. He reviewed the patches and contributed many detail ideas.

Legal Information

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.
- Intel may make changes to specifications, product descriptions, and plans at any time, without notice.
- All dates provided are subject to change without notice.
- Intel is a trademark of Intel Corporation in the U.S. and other countries.
- *Other names and brands may be claimed as the property of others.
-