Dirty throttling — How much dirty memory is too much?

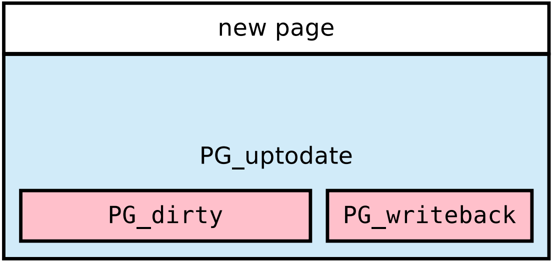
Wu Fengguang
<wfg@linux.intel.com>

October 17, 2010

Outline

- **writeback basics**
- **dirty limits**
- **dirty throttling algorithms**

page cache pages



new page

PG_uptodate

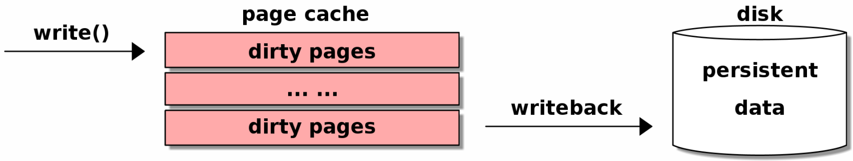
PG_dirty

PG_writeback

```
new page:       no valid data
PG_uptodate:  have valid data
PG_dirty:     have valid data, to be synced to disk
PG_writeback: have valid data, being synced to disk

clean page: PG_uptodate && !PG_dirty && !PG_writeback
dirty page: PG_uptodate && (PG_dirty || PG_writeback)
```

writeback: delaying the write IO



Benefits

- async IO: avoid blocking the apps
- avoid IO (eg. temp files)
- batched IO: better throughput

Question: When to writeback the dirty pages?

option 1: sync syscalls

```
- sync()

- fsync()
- fdatasync()
- sync_file_range()

- msync()

- open(O_SYNC)
- open(O_DIRECT)
```

option 2: the flusher thread(s)

- initiate writeback IO in the background
- one flusher thread per storage device

```
$ ps ax
  PID TTY      STAT   TIME COMMAND
 2322 ?        S      0:01 [flush-8:0]
12681 ?        S      0:00 [flush-btrfs-1]
```

when to writeback

- dirty expire time: 30 seconds

- background flush threshold: 10% memory dirtied
- dirty throttling threshold: 20% memory dirtied

who to initiate IO

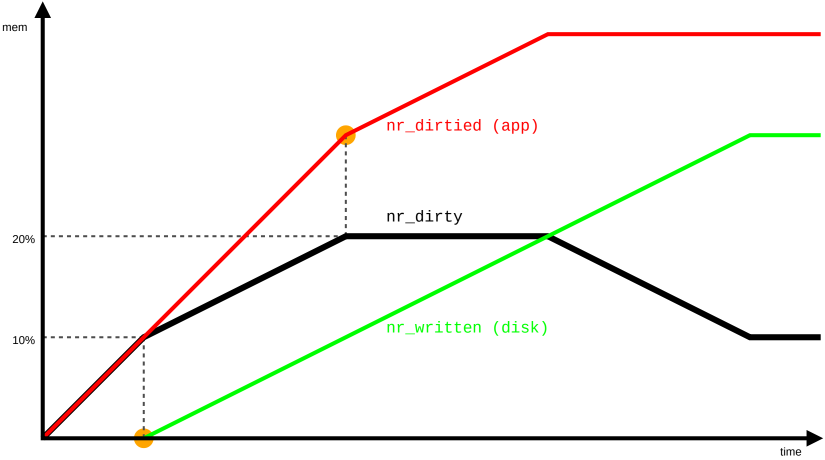
fsync()	the call task	SLOW
sync()	the flusher thread	
periodic writeback	the flusher thread	
background writeback	the flusher thread	

problematic writeback paths

balance_dirty_pages()	the current dirtier	slow
page reclaim	kswapd and/or page allocate task	VERY SLOW

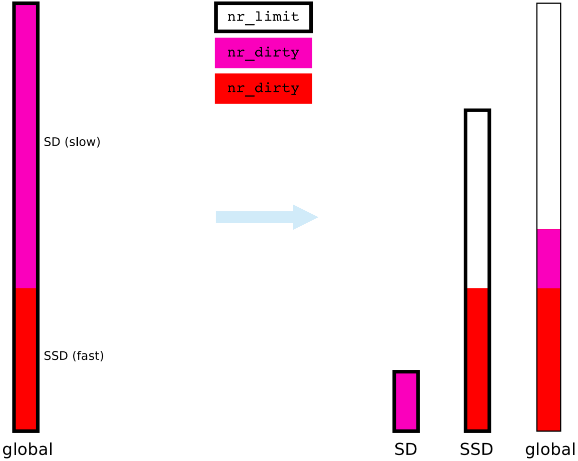
SLOW = IO inefficient + slow responsiveness

dirty limits illustrated



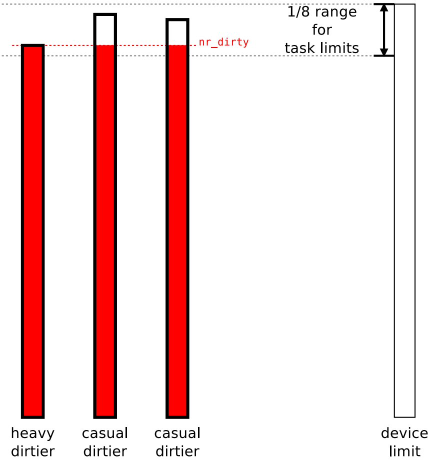
per-device dirty limits

solution for: inter device starvation



per-task dirty limits

solution for: inter process starvation



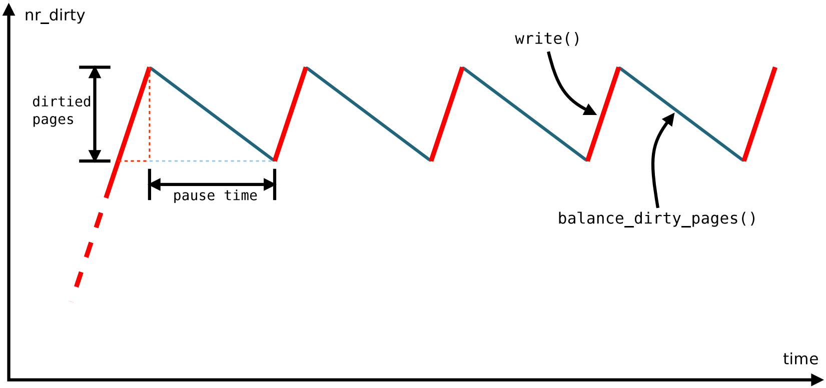
balance_dirty_pages()

```
sys_write()
    balance_dirty_pages()
```

```
        if (task_dirty_exceeded())
                writeback_inodes(dirtied * 3/2);

        if (over_bground_thresh())
                start_background_writeback();
```

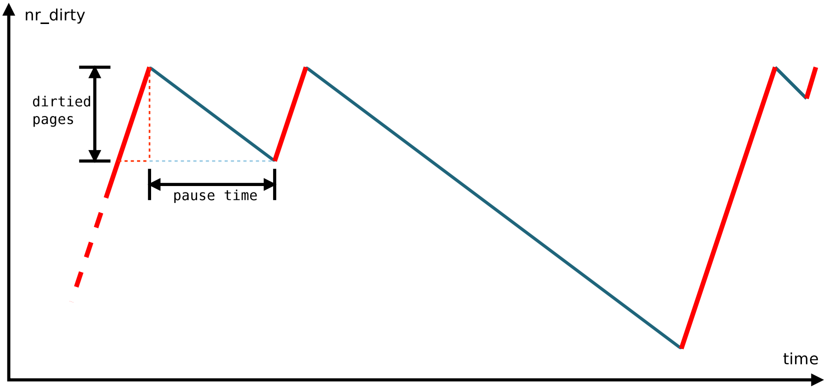
balance_dirty_pages() parameters



Ideal is: (1) dirtied pages < dirty_limit/100 (easy)

(2) 1ms < pause time < 100ms (important)

balance_dirty_pages() is not latency wise



Problems: (1) pause time won't scale to storage speed

(2) pause time fluctuates a lot in one task

balance_dirty_pages() is not IO wise

- **seeky IO**

  ```
  parallel dirtiers
  => N dirtiers working on N inodes
  => interleaved IO to multiple disk regions
  ```

- **small IO size**

  ```
  pause time limit
  => small write size
  => small extent size
  => small read size
  ```

Solution: IO-less `balance_dirty_pages()`

try 1: wait for IO completion

```
        if (task_dirty_exceeded())
-               writeback_inodes(dirtied * 3/2);
+               wait_for_writeback(dirtied * 3/2);
```

- bumpy IO completion on NFS
- accounting inaccuracy and overheads

try 2: sleep for estimated time

```
        if (task_dirty_exceeded())
-               wait_for_writeback(dirtied * 3/2);
+               sleep(dirtied * 3/2 / write_bandwidth);
```

- estimation problem on multiple sleepers
- estimation problem with advanced limits

try 3: sleep for controlled time

```
        if (task_dirty_exceeded())
-               sleep(dirtied * 3/2 / write_bandwidth);
+               sleep(dirtied / throttle_bandwidth);
```

+ directly control pause time

+ convenient for dynamic limit and IO controller

throttle bandwidth

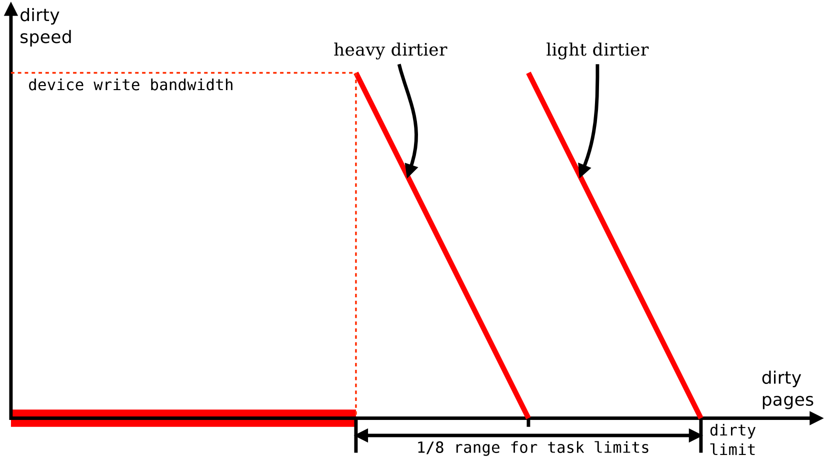
$$\texttt{device\_limit} = \texttt{device\_weight} \times \texttt{global\_limit} \tag{1}$$

$$\texttt{task\_limit} = \texttt{device\_limit} - \texttt{task\_weight} \times \frac{\texttt{device\_limit}}{16} \tag{2}$$

$$\texttt{throttle\_bandwidth} = \texttt{device\_bandwidth} \times \frac{\texttt{task\_limit} - \texttt{nr\_dirty}}{\texttt{task\_limit}/16} \tag{3}$$

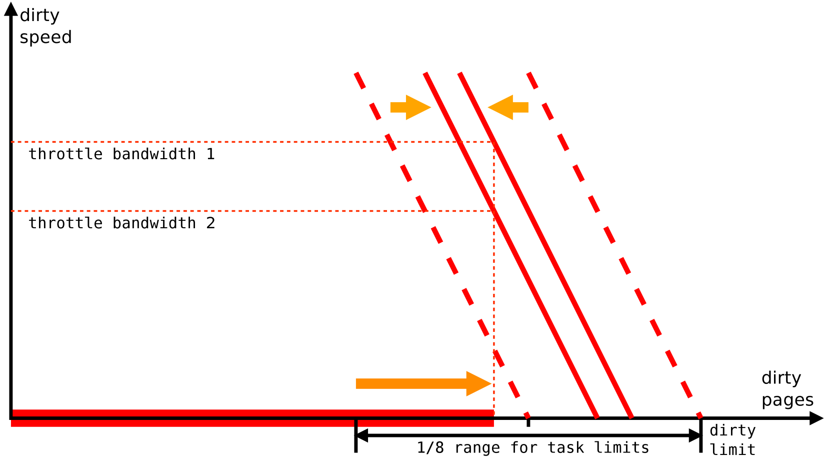
throttle bandwidth (state 1)

heavy dirtier + light dirtier



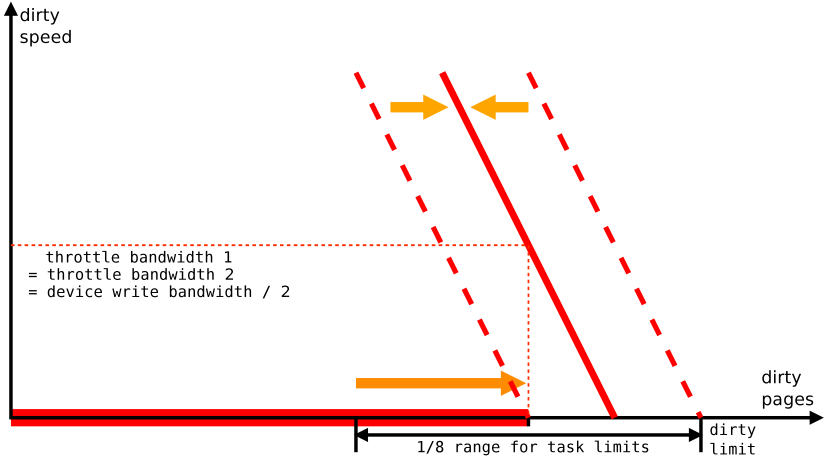
throttle bandwidth (state 2)

light dirtier => heavy dirtier

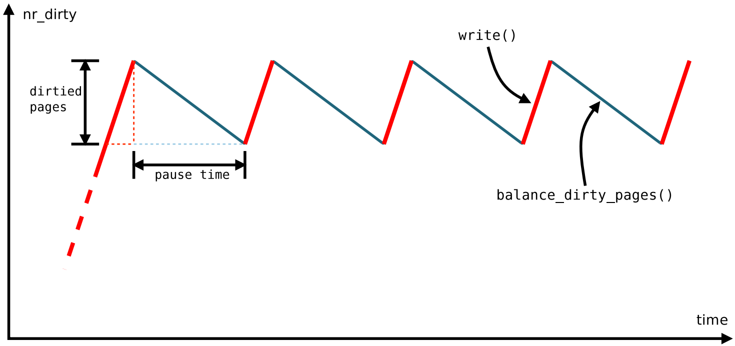


throttle bandwidth (state 3)

stable state: two heavy dirtiers



pause time



```
pause_time = nr_dirtied / throttle_bandwidth

if      (pause_time < 1ms)      max_nr_dirtied += 1;
else if (pause_time > 100ms)    max_nr_dirtied /= 2;
```

dynamic dirty limit: rational

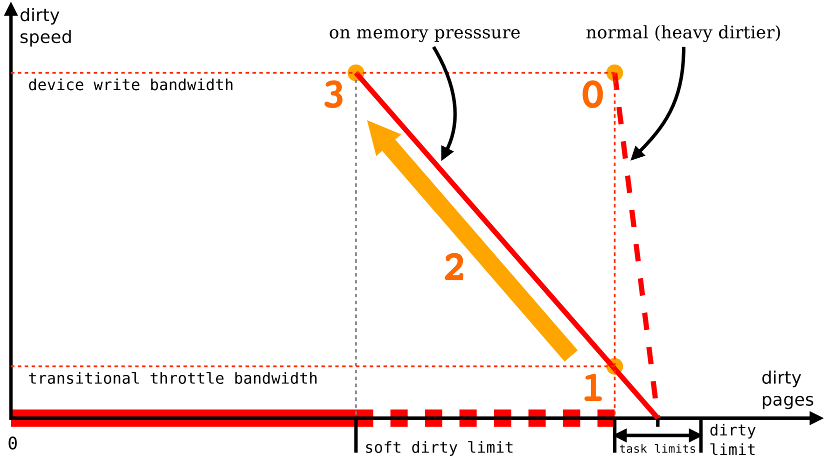
dirty pages may hurt under memory pressure

- **eats** 20% **memory**
- **triggers writeback on page reclaim**
  - 4k seeky IO
  - high latency

```
         LRU list
<======= ---------------------*-----------*--------*-----*-----**----*---*--*-*-
 reclaim     [-] clean page
             [*] dirty page
```

dynamic dirty limit via soft dirty limit

stable state: two heavy dirtiers



what's next

- **per-cgroup dirty limits**
- **write IO controller**

Thank you!

intel® Leap ahead™