linux数据同步机制分析

宝德技术研究院 李磊

# 周期性回写

在start\_kernel中对回写周期进行了设置，当然需要使用定时器。初始化是调用函数page\_writeback\_init实现的

void \_\_init page\_writeback\_init(void)

{

long buffer\_pages = nr\_free\_buffer\_pages();

long correction;

total\_pages = nr\_free\_pagecache\_pages();

correction = (100 \* 4 \* buffer\_pages) / total\_pages;

if (correction < 100) {

dirty\_background\_ratio \*= correction;

dirty\_background\_ratio /= 100;

vm\_dirty\_ratio \*= correction;

vm\_dirty\_ratio /= 100;

if (dirty\_background\_ratio <= 0)

dirty\_background\_ratio = 1;

if (vm\_dirty\_ratio <= 0)

vm\_dirty\_ratio = 1;

}

mod\_timer(&wb\_timer, jiffies + dirty\_writeback\_interval);

set\_ratelimit();

register\_cpu\_notifier(&ratelimit\_nb);

}

这个函数中，我们关系的是下面这行代码

mod\_timer(&wb\_timer, jiffies + dirty\_writeback\_interval);

这显然就要继续跟踪wb\_timer了。这个定时器是一个全局的变量，通过下面代码被初始化。

static DEFINE\_TIMER(wb\_timer, wb\_timer\_fn, 0, 0);

#define TIMER\_INITIALIZER(\_function, \_expires, \_data) { \

.function = (\_function), \

.expires = (\_expires), \

.data = (\_data), \

.base = &boot\_tvec\_bases, \

}

#define DEFINE\_TIMER(\_name, \_function, \_expires, \_data) \

struct timer\_list \_name = \

TIMER\_INITIALIZER(\_function, \_expires, \_data)

通过上面分析，很明显可以看出wb\_timer定时器dirty\_writeback\_interval过后就会激活，然后调用wb\_timer\_fn。而这个dirty\_writeback\_interval默认为5s。再来看看wb\_timer\_fn。这个函数会反复被调用，时间间隔为1s。

static void wb\_timer\_fn(unsigned long unused)

{

if (pdflush\_operation(wb\_kupdate, 0) < 0)

mod\_timer(&wb\_timer, jiffies + HZ); /\* delay 1 second \*/

}

## pdflush\_operation

## wb\_kupdate