

---

Subject: [RFC 6/7] Add min and max statistics to percpu\_counter

Posted by [Glauber Costa](#) on Fri, 30 Mar 2012 08:04:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Because percpu counters can accumulate their per-cpu sums over time outside of the control of the callers, we need this patch that updates the maximum values when that happens.

I am adding a minimum value as well for consistency, due to the signed nature of the percpu\_counters.

However, I am not sure this will be of general use, and might be yet another indication that we need to duplicate those structures...

Signed-off-by: Glauber Costa <[glommer@parallels.com](mailto:glommer@parallels.com)>

---

```
include/linux/percpu_counter.h |  2 ++
include/linux/res_counter.h   |  6 +-----
kernel/res_counter.c         |  6 +-----
lib/percpu_counter.c         |  4 +----
4 files changed, 10 insertions(+), 8 deletions(-)
```

```
diff --git a/include/linux/percpu_counter.h b/include/linux/percpu_counter.h
index 8310548..639d2d5 100644
--- a/include/linux/percpu_counter.h
+++ b/include/linux/percpu_counter.h
@@ -18,6 +18,8 @@
 struct percpu_counter {
     raw_spinlock_t lock;
     s64 count;
+    s64 max;
+    s64 min;
 #ifdef CONFIG_HOTPLUG_CPU
     struct list_head list; /* All percpu_counters are on a list */
 #endif
diff --git a/include/linux/res_counter.h b/include/linux/res_counter.h
index 8c1c20e..3527827 100644
--- a/include/linux/res_counter.h
+++ b/include/linux/res_counter.h
@@ -27,10 +27,6 @@ struct res_counter {
 */
 struct percpu_counter usage_pcp;
 /*
 - * the maximal value of the usage from the counter creation
 - */
- unsigned long long max_usage;
```

```

- /*
 * the limit that usage cannot exceed
 */
unsigned long long limit;
@@ -178,7 +174,7 @@ static inline void res_counter_reset_max(struct res_counter *cnt)
 unsigned long flags;

 raw_spin_lock_irqsave(&cnt->usage_pcp.lock, flags);
- cnt->max_usage = __percpu_counter_sum_locked(&cnt->usage_pcp);
+ cnt->usage_pcp.max = __percpu_counter_sum_locked(&cnt->usage_pcp);
 raw_spin_unlock_irqrestore(&cnt->usage_pcp.lock, flags);
}

diff --git a/kernel/res_counter.c b/kernel/res_counter.c
index 8a99943..7b05208 100644
--- a/kernel/res_counter.c
+++ b/kernel/res_counter.c
@@ -61,8 +61,8 @@ int __res_counter_add(struct res_counter *c, long val, bool fail)

 c->usage_pcp.count += val;

- if (usage > c->max_usage)
- c->max_usage = usage;
+ if (usage > c->usage_pcp.max)
+ c->usage_pcp.max = usage;

out:
 raw_spin_unlock(&c->usage_pcp.lock);
@@ -164,7 +164,7 @@ res_counter_member(struct res_counter *counter, int member)
{
 switch (member) {
 case RES_MAX_USAGE:
- return &counter->max_usage;
+ return &counter->usage_pcp.max;
 case RES_LIMIT:
 return &counter->limit;
 case RES_FAILCNT:
diff --git a/lib/percpu_counter.c b/lib/percpu_counter.c
index 0b6a672..6dff70b 100644
--- a/lib/percpu_counter.c
+++ b/lib/percpu_counter.c
@@ -81,6 +81,10 @@ void __percpu_counter_add(struct percpu_counter *fbc, s64 amount, s32
batch)
 raw_spin_lock(&fbc->lock);
 fbc->count += count;
 __this_cpu_write(*fbc->counters, 0);
+ if (fbc->count > fbc->max)
+ fbc->max = fbc->count;

```

```
+ if (fbc->count < fbc->min)
+   fbc->min = fbc->count;
  raw_spin_unlock(&fbc->lock);
} else {
  __this_cpu_write(*fbc->counters, count);
--
```

#### 1.7.4.1

---