Subject: [RFC 0/7] Initial proposal for faster res_counter updates
Posted by Glauber Costa on Fri, 30 Mar 2012 08:04:38 GMT
View Forum Message <> Reply to Message

Hi,

Here is my take about how we can make res_counter updates faster.
Keep in mind this is a bit of a hack intended as a proof of concept.

The pros I see with this:

* free updates in non-constrained paths. non-constrained paths includes
  unlimited scenarios, but also ones in which we are far from the limit.

* No need to have a special cache mechanism in memcg. The problem with
  the caching is my opinion, is that we will forward-account pages, meaning
  that we'll consider accounted pages we never used. I am not sure
  anyone actually ran into this, but in theory, this can fire events
  much earlier than it should.

But the cons:

* percpu counters have signed quantities, so this would limit us 4G.
  We can add a shift and then count pages instead of bytes, but we
  are still in the 16T area here. Maybe we really need more than that.

* some of the additions here may slow down the percpu_counters for
  users that don't care about our usage. Things about min/max tracking
  enter in this category.

* growth of the percpu memory.

It is still not clear for me if we should use percpu_counters as this
patch implies, or if we should just replicate its functionality.

I need to go through at least one more full round of auditing before
making sure the locking is safe, specially my use of synchronize_rcu().

As for measurements, the cache we have in memcg kind of distort things.
I need to either disable it, or find the cases in which it is likely
to lose and benchmark them, such as deep hierarchy concurrent updates
with common parents.

I also included a possible optimization that can be done when we
are close to the limit to avoid the initial tests altogether, but
it needs to be extended to avoid scanning the percpu areas as well.

In summary, if this is to be carried forward, it definitely needs

some love. It should be, however, more than enough to make the
proposal clear.

Comments are appreciated.

Glauber Costa (7):
  split percpu_counter_sum
  consolidate all res_counter manipulation
  bundle a percpu counter into res_counters and use its lock
  move res_counter_set limit to res_counter.c
  use percpu_counters for res_counter usage
  Add min and max statistics to percpu_counter
  Global optimization

 include/linux/percpu_counter.h | 3 +
 include/linux/res_counter.h | 63 ++++++----------
 kernel/res_counter.c | 151 ++++++++++++++++++++++++++++++----------
 lib/percpu_counter.c | 16 ++++-
 4 files changed, 151 insertions(+), 82 deletions(-)

--
1.7.4.1