

Re: [RFC][PATCH 1/2] memcg: res_counter hierarchy

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-05/msg14180.html>

- *From:* Balbir Singh <balbir@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 31 May 2008 16:50:34 +0530
-

kamezawa.hiroyu@xxxxxxxxxxxxxxxxxx wrote:

KAMEZAWA Hiroyuki wrote:

This patch tries to implements `_simple_` 'hierarchy policy' in `res_counter`.

While several policy of hierarchy can be considered, this patch implements simple one

- the parent includes, over-commits the child
- there are no shared resource

I am not sure if this is desirable. The concept of a hierarchy applies really well when there are shared resources.

- dynamic hierarchy resource usage management in the kernel is not neces

sary

Could you please elaborate as to why? I am not sure I understand your point

ok, let's consider a `_middleware_` which has following paramater.

An expoterd param to the user.

- `user_memory_limit`

parameters for co-operation with the kernel

- `kernel_memory_limit`

And here,

$$\text{user_memory_limit} \geq \text{kernel_memory_limit} == \text{cgroup's memory.limits_in_bytes}$$

When a user ask the miidleware to set limit to 1Gbytes

$$\text{user_memory_limit} = 1\text{G}$$

Re: [RFC][PATCH 1/2] memcg: res_counter hierarchy

kernel_memory_limit = 0-1G.

It moves kernel_memory_limit dynamically 0 to 1Gbytes and reset limits_in_bytes in dynamic way with checking memory cgroup's statistics.

Of course, we can add some kind of interface, as following

- failure_notifier - triggered at failcnt increment.
- threshold_notifier - triggered as usage > threshold.

works as following.

1. create a child. set default child limits to be 0.
2. set limit to child.
 - 2-a. before setting limit to child, prepare enough room in parent.
 - 2-b. increase 'usage' of parent by child's limit.

The problem with this is that you are forcing the parent will run into a reclaim
aim

loop even if the child is not using the assigned limit to it.

That's not problem because it's avoidable by users.

But it's ok to limit the sum of child's limit to be below XX % of the parent.

3. the child sets its limit to the val moved from the parent.
the parent remembers what amount of resource is to the children.

All of this needs to be dynamic

As explained, this can be dynamic by middleware.

Above means that

- a directory's usage implies the sum of all sub directories + own usage.
- there are no shared resource between parent <-> child.

Pros.

- simple and easy policy.
- no hierarchy overhead.
- no resource share among child <-> parent. very suitable for multilevel resource isolation.

Re: [RFC][PATCH 1/2] memcg: res_counter hierarchy

Sharing is an important aspect of hierachies. I am not convinced of this approach. Did you look at the patches I sent out? Was there something fundamentally broken in them?

Yes, I read. And tried to make it faster and found it will be complicated. One problem is overhead of counter itself. Another problem is overhead of shrinking multi-level LRU with feedback. One more problem is that it's hard to implement various kinds of hierarchy policy. I believe there are other hierarhcy policies rather than OpenVZ want to use. Kicking out functions to middleware AMAP is what I'm thinking now.

One way to manage hierarchies other than via limits is to use shares (please see the shares used by the cpu controller). Basically, what you've done with limits is done with shares

If a parent has 100 shares, then it can decide how many to pass on to it's children based on the shares of the child and your logic would work well. I propose assigning top level (high resolution) shares to the root of the cgroup and in a hierarchy passing them down to children and sharing it with them. Based on the shares, deduce the limit of each node in the hierarchy.

What do you think?

--

Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>