Subject: Re: [PATCH 0/3] cgroup: block device i/o bandwidth controller
Posted by Andrea Righi on Sat, 24 May 2008 22:28:06 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Andrea Righi wrote:
>> I'm resending an updated version of the cgroup I/O bandwidth controller
>> that I wrote some months ago (http://lwn.net/Articles/265944), since
>> someone asked me recently.
>>
>> The goal of this patchset is to implement a block device I/O bandwidth
>> controller using cgroups.
>>
>> Detailed informations about design, its goal and usage are described in
>> the documentation.
>>
>> Review, comments and feedbacks are welcome.
>>
>
> Hi, Andrea,
>
> There are several parallel efforts for the IO controller? Could you
> describe/compare them with yours? Is any consensus building taking place?
>
> CC'ing containers mailing list.
>

Hi Balbir,

I've seen the Ryo Tsuruta's dm-band (http://lwn.net/Articles/266257),
the CFQ cgroup solution proposed by Vasily Tarasov
(http://lwn.net/Articles/274652) and a similar approach by Satoshi
Uchida (http://lwn.net/Articles/275944).

First one is implemented at the device mapper layer and AFAIK at the
moment it allows only to define per-task, per-user and per-group rules
(cgroup support is in the TODO list anyway).

Second and third solutions are implemented at the i/o scheduler
layer, CFQ in particular.

They work as expected with direct i/o operations (or synchronous reads).
The problem is: how to limit the i/o activity of an applicaton that
already wrote all the data in memory? The i/o scheduler is not the right
place to throttle application writes, because it's "too late". At the
very least we can map back the i/o path to find which process originally
dirtied memory and depending on this information delay the dispatching
of the requests. However, this needs bigger buffers, more page cache

usage, that can lead to potential OOM conditions in massively shared environments.

So, my approach has the disadvantage or the advantage (depending on the context and the requirements) to explicitly choke applications' requests. Other solutions that operates in the subsystem used to dispatch i/o requests are probably better to maximize overall performance, but do not offer the same control over a real QoS as request limiting can do.

Another difference is that all of them are priority/weighted based. The io-throttle controller, instead, allows to define direct bandwidth limiting rules. The difference here is that priority based approach propagates bursts and does no smoothing. Bandwidth limiting approach controls bursts by smoothing the i/o rate. This means better performance predictability at the cost of poor throughput optimization.

I'll run some benchmarks and post the results ASAP. It would be also interesting to run the same benchmarks using the other i/o controllers and compare the results in terms of fairness, performance predictability, responsiveness, throughput, etc. I'll see what I can do.

-Andrea

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers