Subject: Re: Supporting overcommit with the memory controller
Posted by KAMEZAWA Hiroyuki on Thu, 06 Mar 2008 00:59:31 GMT
View Forum Message <> Reply to Message

On Wed, 5 Mar 2008 16:17:13 -0800
"Paul Menage" <menage@google.com> wrote:
> Users are poor at determining how much memory their jobs will actually
> use (partly due to poor estimation, partly due to high variance of
> memory usage on some jobs). So, we want to overcommit machines, i.e.
> we want the total limits granted to all cgroups add up to more than
> the total size of the machine.
>
just depends on middle-ware. I think most of them will not allow that.

> So for each job we need a (per-job configurable) amount of memory
> that's essentially reserved for that job. That way the high-priority
> job can carry on allocating from its reserved pool even while the
> low-priority job is OOMing; the low-priority job can't touch the
> reserved pool of the high-priority job.
>
Hmm, but current resource charging is independent from page allocator.
(I think this is a good aspect of current design.)

> But to make this more interesting, there are plenty of jobs that will
> happily fill as much pagecache as they have available. Even a job
> that's just writing out logs will continually expand its pagecache
> usage without anything to stop it, and so just keeping the reserved
> pool at a fixed amount of free memory will result in the job expanding
> even if it doesn't need to.
It's current memory management style. "reclaim only when necessary".

> Therefore we want to be able to include in
> the "reserved" pool, memory that's allocated by the job, but which can
> be freed without causing performance penalties for the job. (e.g. log
> files, or pages from a large on-disk data file with little access
> locality of reference) So suppose we'd decided to keep a reserve of
> 200M for a particular job - if it had 200M of stale log file pages in
> the pagecache then we could treat those as the 200M reserve, and not
> have to keep on expanding the reserve pool.
>
> We've been approximating this reasonably well with a combination of
> cpusets, fake numa, and some hacks to determine how many pages in each
> node haven't been touched recently (this is a bit different from the
> active/inactive distinction). By assigning physical chunks of memory
> (fake numa nodes) to different jobs, we get the pre-reservation that
> we need. But using fake numa is a little inflexible, so it would be
> nice to be able to use a page-based memory controller.

>
> Is this something that would be possible to set up with the current
> memory controller? My impression is that this isn't quite possible
> yet, but maybe I've not just thought hard enough. I suspect that we'd
> need at least the addition of page refault data, and the ability to
> pre-reserve pages for a group.
>
Can Balbir's soft-limit patches help ?

It reclamims each cgroup's pages to soft-limit if the system needs.

Make limitation like this

Assume 4G server.

	Limit	soft-limit
Not important Apss:	2G	100M
Important Apps :	3G	2.7G

When the system memory reachs to the limit, each cgroup's memory usages will goes down to soft-limit. (And there will 1.3G of free pages in above example)

Thanks,
-Kame