

Re: excessive swap-in time

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.system/2008-03/msg00143.html>

- *From:* phil-news-nospam@xxxxxxxx
 - *Date:* 19 Mar 2008 14:05:07 GMT
-

On Tue, 18 Mar 2008 20:02:33 -0700 (PDT) David Schwartz <davids@xxxxxxxxxxxxxx> wrote:

| On Mar 18, 6:40 am, phil-news-nos...@xxxxxxxx wrote:

|
|> | If you have no swap space, the system cannot swap out pages that
|> | haven't been touched for a very, very long time. This means that not
|> | only must the working set fit in memory, but all modified pages that
|> | are not part of the working set must fit as well. Thus the system may
|> | page badly even if the working set is less than the total size of
|> | physical memory.

|
|> But it would be no worse with an 8GB RAM / 0GB SWAP system as compared to
|> a 4GB RAM / 4GB SWAP system, or a 2GB RAM / 6GB SWAP.

|
| Absolutely, but swap is orders of magnitude cheaper than RAM. For
| almost any given cost level, you can do better with some of that money
| going to swap. Trying to do it all with RAM is just inefficient.

I find that does not work well when the swapping is caused by excessive I/O buffers staying in RAM. In such cases, the demand on the RAM exceeds the normal VM profile, causing heavy swapping where otherwise there would be none at all. Note that this is a case quite different than the one I started this thread about (which is a fully idle swapped out process coming back in but at the same time causing itself to be swapped out and back in over and over due to poor page stealing selection).

Ideally, once I/O buffer usage reaches a certain threshold, that must be tunable by the administrator at any time, and further demand for an I/O buffer must be met only by taking away another I/O buffer. If all I/O buffers are currently dirty, as would be the case during heavy writing, the process doing the writing must block in write() until a buffer becomes available. The mechanism to block it exists because it has to be prepared to handle fully exhausting memory. What we need is the ability to set a limit on it which effectively reserves other RAM to be used exclusively for paging purposes.

During cases of heavy writing, swapping activity now competes with the I/O activity. The writing could complete faster if the swapping were prevented. So disregarding the issue of keeping other processes runnable, preventing the swapping from taking place (by restricting the buffering) actually makes

Re: excessive swap-in time

the I/O go faster, increasing overall system performance.

I/O is buffered for several reasons. One is to be sure the process is not delayed by the turnarounds back and forth between running a process and writing some data. I/O done fully synchronous would be an example of the case normally preferred to be avoided. By allowing the process to continue running after a write() call, we eliminate the turnaround delays.

Additional I/O speed improvements happen by having sufficient buffers in RAM, in the case where I/O positions are "random" (rather than sequential) which allows the classic elevator algorithm, or other improvements to be performed.

If the buffers being written also need to be read back, by either the same process, or by another one, keeping them in RAM, even if not dirty, helps the performance. However, if the amount of data to write exceeds the I/O buffering capacity, little or no benefit of this type can be realized.

> The way I am doing

> this is to decide what my RAM needs are, what my swap needs are, and make
> the RAM really be the sum of that. Thus those pages that haven't been
> touched for a very, very long time and are stuck in RAM because of no swap
> are not occupying the portion of RAM that was originally considered to be
> the need.

| You will still do more disk I/O than you need to because some pages
| will stuck in memory when it's more efficient for them not to be. So
| even given that, you will lose efficiency because the disk cache will
| be smaller.

> What if I reserve 4GB of that 8GB of RAM to be a ramdisk and use it for
> swapping? What I would be doing is taking those long untouched pages and
> moving them from one place in RAM to another place in RAM. Well ... what
> I would be doing by leaving all 8GB as available RAM is avoiding any page
> movement at all. So those long untouched pages just sit where they are
> and don't take up I/O bandwidth.

| Except that other pages do take up I/O bandwidth because memory is
| wasted holding pages that were modified long ago and won't be read for
| ages — memory that could hold pages that were discarded.

> I can see the two counter arguments here:

>
> 1. Add some swap to the 8GB and you can make even better use of the 8GB
> by having more of the 8GB available by getting those long untouched
> pages out of the way.

| Exactly.

> 2. (mine) Add more RAM instead of adding more swap, and to the extent

Re: excessive swap-in time

|> there are not too many of those long untouched pages, you have more
|> RAM to use.

|

| Crazy, since the cost of adding RAM is so much higher than the cost of
| adding swap.

If you need to add 8GB more because they do not perform well with what you have now, you have to decide if that shall be 8GB of swap or 8GB of RAM. I suggest that the latter (choose RAM) always will perform better. The issue is cost. But if the performance is greatly needed, the cost can be justified.

So far, with my system which was built with excess RAM and uses no swap, I have found it to actually be running much faster and smoother. I still need to do a lot more tests.

Consider the the issue this way. You have a certain amount of money to spend to get the maximum performance. For \$1000 you could have 16 GB of RAM. Or you could have 5 TB of swap space.

Of course the above decision is absurd. It's meant to be an example that swap space is (can be) so plentiful that it is essentially free. So I suggest that making an economic decision between RAM and swap space does not even make sense. I suggest to get as much RAM as you can afford, and use swap space if you need it (and to NOT use it at all if you don't as the kernel can impact your I/O bus and slow things down the way it still implements VM).

|> One problem I have seen, and described elsewhere, is I/O cache/buffer space
|> tends to get in the way too easily in Linux's VM design. Eliminating swap
|> space can reduce that effect. Dirty pages have nowhere to go and so they
|> will not contribute to the I/O load caused by swapping (which in turn slows
|> the real I/O which can cause a backup of I/O buffers in heavy write cases).
|> Undirty text pages would still be stealable. If stolen, they have to be
|> read back in from their file mapping when needed again. But at least this
|> is less I/O load. The idea of preloading `/usr/lib/s/bin` into ramdisk
|> with even more RAM in the machine would get this I/O load way down.

|

| Huh? Your logic doesn't make sense. Evicting long-ago used dirty pages
| is much better than evicting recently-used clean pages.

Except that what I am describing does not involve as much evicting.

|> The arguments come down to "make better use of the RAM you have" vs. "add
|> more RAM when you need it". They tend to be orthogonal arguments. One can
|> go back and forth between them.

|

| That you can add more RAM when you need it is no reason to make less
| than ideal use of the RAM you have.

Re: excessive swap-in time

Re: excessive swap-in time

If we could:

1. Correctly identify pages that won't be needed for a long time in the future (those that have not been used for a long time in the past are an estimate of probability for this)
2. Ensure that excessive outswapping caused by heavy I/O, especially heavy writing, does not happen
3. Ensure a process swapping back in after dormancy does not cause some of itself to swap out (assuming the working set can fit)

Then we would do well with less RAM and more swap. And I for one would like to see the kernel reach this utopia.

> I have used the following criteria in the past to decide how much swap space
> to use in a system: Determine how much time you can tolerate for an idle
> program to become fully active again. Make your swap space size be how much
> can be read in during that much time. For a system in which I would like to
> eliminate such delays, that suggests very little swap. But there is that
> issue of text pages that won't play in the swap space to contend with.

|
| That's ass backwards. That's like saying you should decide how many
| lifejackets to put on a boat based on the number of passengers less
| the maximum number of casualties you can accept. Lifejackets are
| cheap.

But lifejackets are not the same as swap space.

The objective is to get the kernel to do LESS swapping, not more. That can be influenced by how much or how little swap space is available. The boat analogy has no equivalent since having fewer lifejackets does not help keep the boat afloat.

Sure, the processes that are completely idle should get out of RAM and let the active processes make maximum use of it. But that is not how the kernel currently behaves.

I've written for a few years about how the kernel performs poorly when a process is doing a lot of heavy writing that causes outswapping, which in turn slows down the writing. Going from 2.2 to 2.4 actually made this worse. Going from 2.4 to 2.6 improved it some, but not adequately. My past suggestion of having the ability to partition RAM and designate the valid uses of it (specifically to mark some not to be used for write buffers) I still believe would be an administrative way to deal with the issue. I suggest the (costly) workaround of trading swap for RAM.

I've now recently identified the POSSIBILITY that the kernel is making poor selection of pages to steal for inswapping activities (e.g. it

Re: excessive swap-in time

Re: excessive swap-in time

chooses pages of the process it is trying to bring back in, causing it to thrash against itself, delaying its time to fully active). Trading swap for RAM helps get around this as well.

--

|-----/-----|
| Phil Howard KA9WGN (ka9wgn.ham.org) / Do not send to the address below |
| first name lower case at ipal.net / spamtrap-2008-03-19-0823@xxxxxxxx |
|-----/-----|

.