Slabs, Pages, Chunks and Memcached

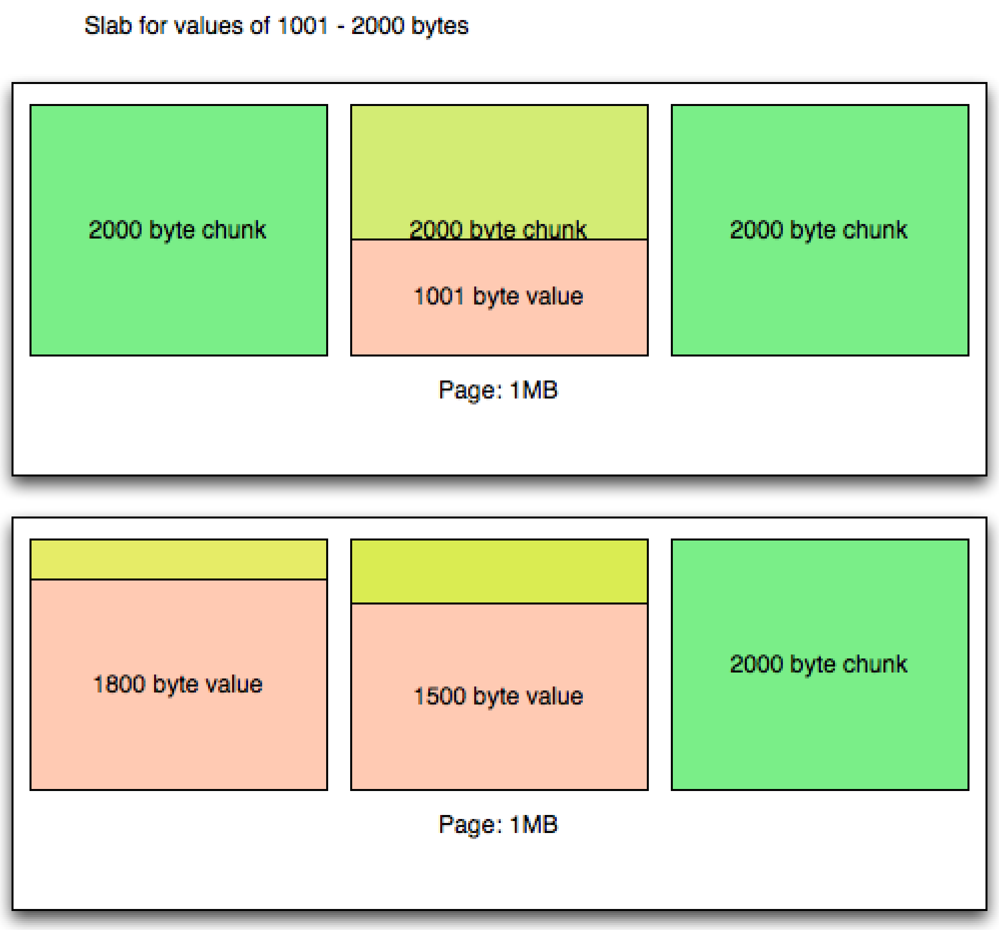
2009-06-22

Many people don't know this but the latest memcached release (1.2.8 right now) can be about 15% more efficient in its memory usage than older releases. If you have a 600MB memcached server, upgrading will magically "gain" you 100MB of RAM. Why is this?

When you ask memcached to store a value, it looks up the "slab" associated with that value. A slab holds values within a particular size range. Slabs are composed of 1MB pages, which are broken into chunks of the slab's size. Let's say your value is 1001 bytes; memcached will look up the slab which holds values between 1000 and 2000 bytes. It then finds a page with an empty chunk and inserts the value into that chunk. Note that a chunk is fixed in size – it must be 2000 bytes in order to store the largest value for the slab.

Now you know why memcached limits values to one megabyte: the value must be stored in a chunk and a page needs to hold the chunk. Since a page is hardcoded as 1MB, it follows that a chunk must also be limited to 1MB.

So we understand the "object model" for memcached memory allocation: a slab has many pages which has many chunks. Each chunk is a fixed size, based on the maximum size for the slab so e.g. the 2000 byte slab will hold values between 1001 and 2000 bytes. Older versions of memcached used slabs sized based on powers of two, so you'd have a 1KB slab, 2KB slab, 4KB slab, …, all the way to 1MB. If your memcached server was full of 1001 byte values, your memory efficiency would be 50% (1001 / 2000) in the worst case. Assuming you have an even distribution of value sizes, you'll get 75% efficiency (1500 / 2000). Your 600MB memcached server will only hold 450MB of actual data!



In this image, we see a single slab with two pages. Each page has several chunks, the green chunks are empty and some have orange values. The yellow area is the waste we are talking about.

One of the improvements Facebook made to memcached last year was moving to a smaller exponential so there is not as much waste in storing values in chunks. Instead of 2^n for the slab allocation, the latest versions of memcached use a much smaller growth exponential, 1.25^n, so you will see slabs with sizes 1KB, 1.25KB, 1.56KB, etc… This means that instead of 25% waste on average, you should see closer to 10%. Effectively you regain 15% of your memcached memory just by installing the latest version!