

zsmalloc

This allocator is designed for use with zram. Thus, the allocator is supposed to work well under low memory conditions. In particular, it never attempts higher order page allocation which is very likely to fail under memory pressure. On the other hand, if we just use single (0-order) pages, it would suffer from very high fragmentation -- any object of size `PAGE_SIZE/2` or larger would occupy an entire page. This was one of the major issues with its predecessor (xvmalloc).

To overcome these issues, zsmalloc allocates a bunch of 0-order pages and links them together using various 'struct page' fields. These linked pages act as a single higher-order page i.e. an object can span 0-order page boundaries. The code refers to these linked pages as a single entity called zpage.

For simplicity, zsmalloc can only allocate objects of size up to `PAGE_SIZE` since this satisfies the requirements of all its current users (in the worst case, page is incompressible and is thus stored "as-is" i.e. in uncompressed form). For allocation requests larger than this size, failure is returned (see `zs_malloc`).

Additionally, `zs_malloc()` does not return a dereferenceable pointer. Instead, it returns an opaque handle (unsigned long) which encodes actual location of the allocated object. The reason for this indirection is that zsmalloc does not keep zspages permanently mapped since that would cause issues on 32-bit systems where the VA region for kernel space mappings is very small. So, before using the allocating memory, the object has to be mapped using `zs_map_object()` to get a usable pointer and subsequently unmapped using `zs_unmap_object()`.

stat

With `CONFIG_ZSMALLOC_STAT`, we could see zsmalloc internal information via `/sys/kernel/debug/zsmalloc/<user name>`. Here is a sample of stat output:

```
# cat /sys/kernel/debug/zsmalloc/zram0/classes

class  size almost_full almost_empty obj_allocated  obj_used pages_used pages_per_zpage
...
...
 9   176           0           1           186           129           8           4
10   192           1           0          2880          2872          135           3
11   208           0           1           819           795           42           2
12   224           0           1           219           159           12           4
...
...
```

class
 index
size
 object size zpage stores
almost_empty
 the number of `ZS_ALMOST_EMPTY` zspages(see below)
almost_full
 the number of `ZS_ALMOST_FULL` zspages(see below)
obj_allocated
 the number of objects allocated
obj_used
 the number of objects allocated to the user
pages_used
 the number of pages allocated for the class
pages_per_zpage
 the number of 0-order pages to make a zpage

We assign a zpage to `ZS_ALMOST_EMPTY` fullness group when $n \leq N / f$, where

- n = number of allocated objects
- N = total number of objects zpage can store
- f = `fullness_threshold_frac`(ie, 4 at the moment)

Similarly, we assign zpage to:

- `ZS_ALMOST_FULL` when $n > N / f$
- `ZS_EMPTY` when $n == 0$
- `ZS_FULL` when $n == N$