Code Commentary On The Linux Virtual Memory Manager

Mel Gorman

July 9, 2007

Contents

List of Figures

List of Tables

Chapter 1

Boot Memory Allocator

Chapter 2

Physical Page Management

Chapter 3

Non-Contiguous Memory Allocation

Chapter 4

Slab Allocator

```
kmem_cache_create(const char *name, size_t size,
     size_t offset, unsigned long flags,
     void (*ctor)(void*, kmem_cache_t *, unsigned long),
     void (*dtor)(void*, kmem_cache_t *, unsigned long))
```
Creates a new cache and adds it to the cache chain

```
kmem_cache_reap(int gfp_mask)
```
Scans at most `REAP_SCANLEN` caches and selects one for reaping all per-cpu objects and free slabs from. Called when memory is tight

```
kmem_cache_shrink(kmem_cache_t *cachep)
```
This function will delete all per-cpu objects associated with a cache and delete all slabs in the `slabs_free` list. It returns the number of pages freed.

```
kmem_cache_alloc(kmem_cache_t *cachep, int flags)
```
Allocate a single object from the cache and return it to the caller

```
kmem_cache_free(kmem_cache_t *cachep, void *objp)
```
Free an object and return it to the cache

```
kmalloc(size_t size, int flags)
```
Allocate a block of memory from one of the sizes cache

```
kfree(const void *objp)
```
Free a block of memory allocated with kmalloc

```
kmem_cache_destroy(kmem_cache_t * cachep)
```
Destroys all objects in all slabs and frees up all associated memory before removing the cache from the chain

Table 4.1: Slab Allocator API for caches

Chapter 5

Process Address Space

Chapter 6

High Memory Management

Chapter 7

Page Frame Reclamation

Chapter 8

Swap Management