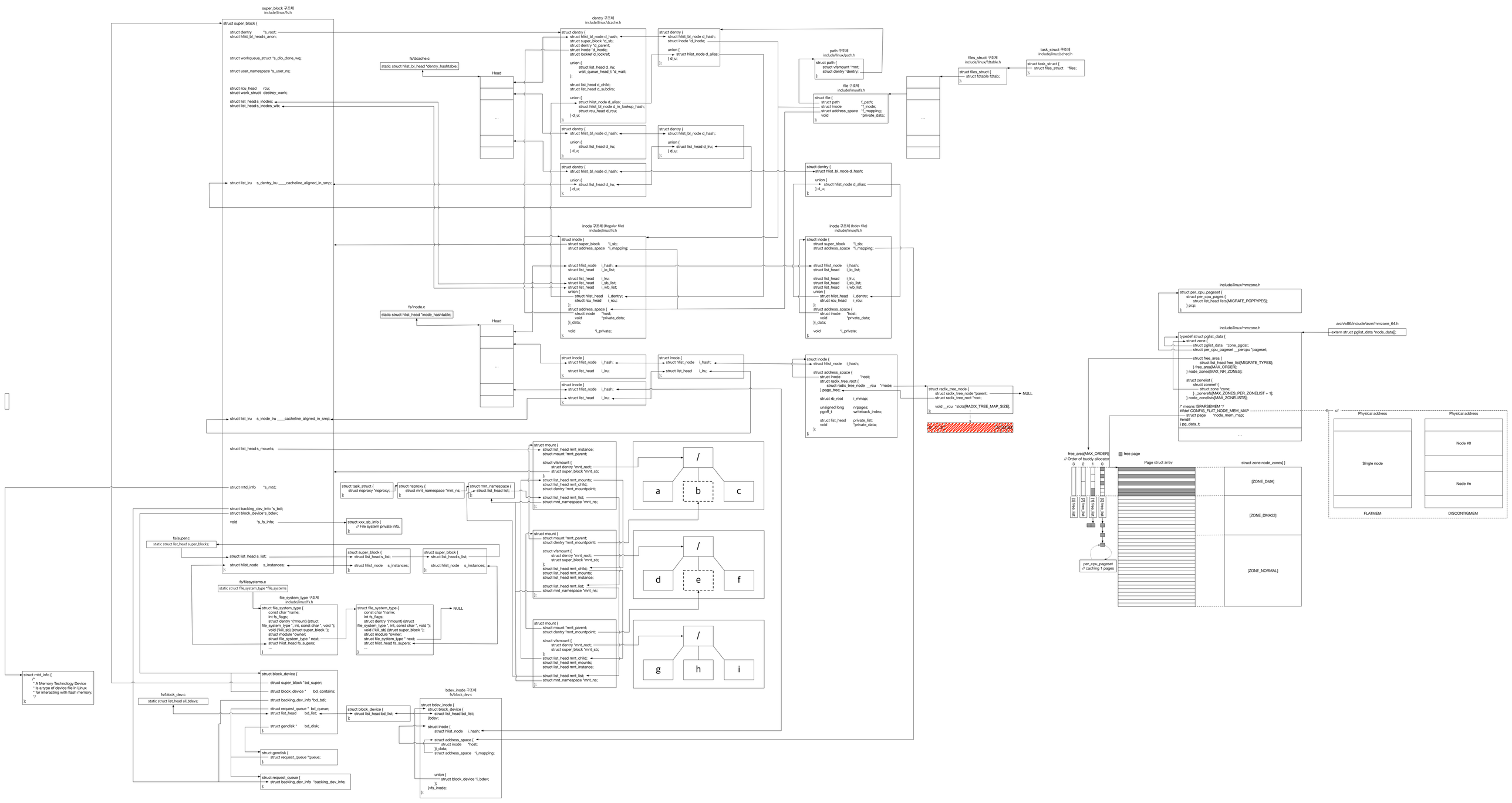
super_block 구조체
include/linux/fs.h
struct super_block {
struct dentry *s_root;
struct hlist_bl_heads_anon;
struct workqueue_struct *s_dio_done_wq;
struct user_namespace *s_user_ns;
struct rcu_head rcu;
struct work_struct destroy_work;
struct list_head s_inodes;
struct list_head s_inodes_wb;
struct list_lru s_dentry_lru ____cacheline_aligned_in_smp;
struct list_lru s_inode_lru ____cacheline_aligned_in_smp;
struct list_head s_mounts;
struct mtd_info *s_mtd;
struct backing_dev_info *s_bdi;
struct block_device*s_bdev;
void *s_fs_info;
dentry 구조체
include/linux/dcache.h
fs/dcache.c
static struct hlist_bl_head *dentry_hashtable;
Head
struct dentry {
struct hlist_bl_node d_hash;
struct super_block *d_sb;
struct dentry *d_parent;
struct inode *d_inode;
struct lockref d_lockref;
union {
struct list_head d_lru;
wait_queue_head_t *d_wait;
};
struct list_head d_child;
struct list_head d_subdirs;
union {
struct hlist_node d_alias;
struct hlist_bl_node d_in_lookup_hash;
struct rcu_head d_rcu;
} d_u;
};
path 구조체
include/linux/path.h
struct path {
struct vfsmount *mnt;
struct dentry *dentry;
};
file_struct 구조체
include/linux/fdtable.h
struct files_struct {
struct fdtable fdtab;
};
task_struct 구조체
include/linux/sched.h
struct task_struct {
struct files_struct *files;
};
file 구조체
include/linux/fs.h
struct file {
struct path f_path;
struct inode *f_inode;
struct address_space *f_mapping;
void *private_data;
};
inode 구조체 (Regular file)
include/linux/fs.h
struct inode {
struct super_block *i_sb;
struct address_space *i_mapping;
struct hlist_node i_hash;
struct list_head i_io_list;
struct list_head i_lru;
struct list_head i_sb_list;
struct list_head i_wb_list;
union {
struct hlist_head i_dentry;
struct rcu_head i_rcu;
};
struct address_space {
struct inode *host;
void *private_data;
} i_data;
void *i_private;
};
inode 구조체 (bdev file)
include/linux/fs.h
fs/inode.c
static struct hlist_head *inode_hashtable;
struct address_space {
struct inode *host;
struct radix_tree_root {
struct radix_tree_node __rcu *mode;
} page_tree;
struct rb_root i_mmap;
unsigned long nrpages;
pgoff_t writeback_index;
struct list_head private_list;
void *private_data;
};
struct radix_tree_node {
struct radix_tree_node *parent;
struct radix_tree_root *root;
void __rcu *slots[RADIX_TREE_MAP_SIZE];
};
NULL
struct task_struct {
struct nsproxy *nsproxy;
};
struct nsproxy {
struct mnt_namespace *mnt_ns;
};
struct mnt_namespace {
struct list_head list;
};
struct mount {
struct list_head mnt_instance;
struct mount *mnt_parent;
struct vfsmount {
struct dentry *mnt_root;
struct super_block *mnt_sb;
};
struct list_head mnt_mounts;
struct list_head mnt_child;
struct dentry *mnt_mountpoint;
struct list_head mnt_list;
struct mnt_namespace *mnt_ns;
};
/
a
b
c
d
e
f
g
h
i
struct xxx_sb_info {
// File system private info.
};
fs/super.c
static struct list_head super_blocks;
struct super_block {
struct list_head s_list;
struct hlist_node s_instances;
};
fs/filesystems.c
static struct file_system_type *file_systems
file_system_type 구조체
include/linux/fs.h
struct file_system_type {
const char *name;
int fs_flags;
struct dentry *(*mount) (struct file_system_type *, int, const char *, void *);
void (*kill_sb) (struct super_block *);
struct module *owner;
struct file_system_type * next;
struct hlist_head fs_supers;
...
}
struct mtd_info {
/*
* A Memory Technology Device
* is a type of device file in Linux
* for interacting with flash memory.
*/
};
fs/block_dev.c
static struct list_head all_bdevs;
struct block_device {
struct super_block *bd_super;
struct block_device * bd_contains;
struct backing_dev_info *bd_bdi;
struct request_queue * bd_queue;
struct list_head bd_list;
struct gendisk * bd_disk;
};
struct gendisk {
struct request_queue *queue;
};
struct request_queue {
struct backing_dev_info *backing_dev_info;
};
bdev_inode 구조체
fs/block_dev.c
struct bdev_inode {
struct block_device {
struct list_head bd_list;
}bdev;
struct inode {
struct hlist_node i_hash;
};
struct address_space {
struct inode *host;
}i_data;
struct address_space *i_mapping;
union {
struct block_device *i_bdev;
};
}vfs_inode;
include/linux/mmzone.h
struct per_cpu_pageset {
struct per_cpu_pages {
struct list_head lists[MIGRATE_PCPTYPES];
} pcp;
};
include/linux/mmzone.h
typedef struct pglist_data {
struct zone {
struct pglist_data *zone_pgdat;
struct per_cpu_pageset __percpu *pageset;
struct free_area {
struct list_head free_list[MIGRATE_TYPES];
} free_area[MAX_ORDER];
} node_zones[MAX_NR_ZONES];
struct zonelist {
struct zoneref {
struct zone *zone;
} _zonerefs[MAX_ZONES_PER_ZONELIST + 1];
} node_zonelists[MAX_ZONELISTS];
/* means ISPARSEMEM */
#ifdef CONFIG_FLAT_NODE_MEM_MAP
struct page *node_mem_map;
#endif
} pg_data_t;
arch/x86/include/asm/mmzone_64.h
extern struct pglist_data *node_data[];
free_area[MAX_ORDER]
// Order of buddy allocator
3 2 1 0
free page
Page struct array
struct zone node_zones[ ]
[ZONE_DMA]
[ZONE_DMA32]
[ZONE_NORMAL]
per_cpu_pageset
// caching 1 pages
cf
Physical address
Physical address
Single node
Node #0
Node #n
FLATMEM
DISCONTIGMEM