Solaris Volume Manager : Kernel Data Structures

md_set

- Array of md_set_t structures
- Array allocated during init of md driver
- Contains status and shortcuts

```
>::sizeof md_set_t
sizeof (md_set_t) = 0x70

X == set you want to look at

> md_set+(X*70)::print md_set_t
{
        s_status = 0x12             -> set status is important
        s_ui = 0x60003400000        -> array unit incore
        s_un = 0x600032e8000        -> array of units
        s_hsp = 0                   -> ptr to hot spare pools
        s_hs = 0                    -> ptr to hotspares
        s_db = 0x60005d94a80        -> ptr to mddb
```

```
    s_dbmx = {
        _opaque = [ 0 ]
    }
    s_nm = 0x60005df7e80          -> ptr to namespace
    s_nmid = 0x4000001                -> 1st namespace record id
    s_did_nm = 0x60005cd0e10 -> ptr to did namespace
    s_did_nmid = 0x4000004            -> 1st namespace did rec id
    s_dtp = 0x60005d6c600         -> data tag ptr
    s_am_i_master = 0             -> multi-owner diskset master
    s_nodeid = 0                  -> multi-owner node id
    s_rcnt = 0                    -> multi-owner resync cnt used
}                                 to balance resync across multiple
                                  nodes
```

s_ui

- s_ui is an array [8192] of pointers to unit incore structs mdi_unit_t
  - ui_readercnt (number of readers)
  - ui_wanabecnt (number of waiting writers)
  - ui_lock
  - ui_ocnt (shows if/how a device is opened)

s_un

- s_un is a corresponding array[8192] of pointers to unit structs
  - md_unit_t – header (mdc_unit) common to all
  - mm_unit_t – mirror
  - mr_unit_t – raid
  - mp_unit_t – softpart
  - ms_unit_t - stripe

s_hsp

- s_hsp is a linked list of hot spare pool information for this set
  - hsp_refcount – number of metadevices using hsp
  - hsp_nhotspares – number of hotspares in pool
  - hsp_hotspares – array of recids for hotspare devices

s_hs

- s_hs is a linked list of all hotspare devices for the set
  - Used to quickly find hotspare device given recid

Looking at unit structures

```
> md_set+(70)::print md_set_t  s_un
s_un = 0x600032e8000

> 0x600032e8000/4J
0x600032e8000:          0     600054b8e00     0          0
```

Unit 1 exists. The array is 8192 long, but I only showed 4.

Looking at unit 1 structure

- Use generic unit structures (md_unit) at first
- un_type can be:
  - MD_DEVICE (stripe)
  - MD_METAMIRROR
  - MD_METATRANS
  - MD_METARAID
  - MD_METASP (softpart)

Looking at Unit 1 Structure

```
>600054b8e00::print md_unit_t
{
    c = {
        un_revision = 0
        un_type = 2 (MD_METAMIRROR)
        un_status = 0
        un_parent_res = 0
        un_child_res = 0
        un_self_id = 0x2001
        un_record_id = 0x4000008
        un_size = 0x210
```

Looking at Unit 1 Structure (cont)

```
        un_flag = 0
        un_total_blocks = 0x88b2e80
        un_actual_tb = 0x88b2e80
        un_nhead = 0x18
        un_nsect = 0x1a8
        un_rpm = 0x2729
        un_wr_reinstruct = 0x11b
        un_rd_reinstruct = 0
        un_vtoc_id = 0
        un_capabilities = 0xd
        un_parent = 0xffffffff
        un_user_flags = 0
    }
}
```

Mirror Specific Unit Structure

```
60054b8e00::print mm_unit_t
{
    c = {
        un_revision = 0
        un_type = 2 (MD_METAMIRROR)
        un_status = 0
        un_parent_res = 0
        un_child_res = 0
        un_self_id = 0x2001
        un_record_id = 0x4000008    /* record 8 in diskset 1 */
        un_size = 0x210
        un_flag = 0
        un_total_blocks = 0x88b2e80
```

Mirror Specific Unit Structure (cont)

```
        un_actual_tb = 0x88b2e80
        un_nhead = 0x18
        un_nsect = 0x1a8
        un_rpm = 0x2729
        un_wr_reinstruct = 0x11b
        un_rd_reinstruct = 0
        un_vtoc_id = 0
        un_capabilities = 0xd
        un_parent = 0xffffffff          /* no parent – top level */
        un_user_flags = 0
}
un_last_read = 0
un_changecnt = 0
```

Mirror Specific Unit Structure (cont)

```
un_nsm = 0x1                /* number of active submirrors */
un_sm = [                   /* array [4] of mm_submirror */
    {
        sm_key = 0x2
        sm_dev = 0x550000200f          /* devt of stripe */
        sm_state = 0x1
        sm_flags = 0x 2
        sm_shared = {
            ms_flags = 0
            ms_state = 0
            ms_lasterrcnt = 0
            ms_orig_dev = 0          /* used by hotspare */
            ms_orig_blk = 0
```

Mirror Specific Unit Structure (cont)

```
            ms_hs_key = 0
            ms_hs_id = 0
            ms_timestamp = {
                tv_sec = 0
                tv_usec  = 0
            }
        }
        sm_hsp_id = 0
        sm_timestamp = {
            tv_sec = 0x43ccc8f3
            tv_usec = 0x7d9cd
        }
}
```

Mirror Specific Unit Structure (cont)

```
}
{                       /* All 0s, no submirror */
        sm_key = 0
        sm_dev = 0
        sm_state = 0
        sm_flags = 0
        sm_shared = {
                ms_flags = 0
                ms_state = 0
                ms_lasterrcnt = 0
                ms_orig_dev = 0
                ms_orig_blk = 0
                ms_hs_key = 0
```

Mirror Specific Unit Structure (cont)

```
            ms_hs_id = 0
            ms_timestamp = {
                tv_sec = 0
                tv_usec = 0
            }
        }
        sm_hsp_id = 0
        sm_timestamp = {
            tv_sec = 0
            tv_usec = 0
        }
    }
```

Mirror Specific Unit Structure (cont)

```
{
        sm_key = 0
        sm_dev = 0
        sm_state = 0
        sm_flags = 0
        sm_shared = {
                ms_flags = 0
                ms_state = 0
                ms_lasterrcnt = 0
                ms_orig_dev = 0
                ms_orig_blk = 0
                ms_hs_key = 0
                ms_hs_id = 0
```

Mirror Specific Unit Structure (cont)

```
                ms_timestamp = {
                    tv_sec = 0
                    tv_usec = 0
                }
            }
            sm_hsp_id = 0
            sm_timestamp = {
                tv_sec =0
                tv_usec = 0
            }
        }
    }
    un_ovrlap_chn_flg = 0
```

Mirror Specific Unit Structure (cont)

```
un_read_option = 0        (RD_LOAD_BAL)
un_write_option = 0       (WR_PARALLEL)
un_pass_num = 0x1
un_resync_flg = 0
un_waiting_to_mark = 0
un_waiting_to_commit = 0
un_rrd_blksize = 0x22feb
un_rrd_num = 0x3e9
un_rr_dirty_recid = 0x4000009        /* opt resync record. Rec 9 in set 1 */
un_rs_copysize = 0
un_rs_dests = 0
un_rs_resync_done = 0                /* amount of resync done. Allows aborted
                                        resync to be restarted without having to
                                        start the resync over. */
```

Mirror Specific Unit Structure (cont)

```
un_rs_resync_2_do = 0          /* amount of total resync */
un_rs_dropped_lock = 0
un_rs_type = 0                 OPTIMIZED, COMPONENT,
                               SUBMIRROR or ABR
un_smic = [
    {
        sm_shared_by_blk = stripe_shared_by_blk
        sm_shared_by_indx = stripe_shared_by_indx
        sm_get_component_count = stripe_component_count
        sm_get_bcss = stripe_block_count_skip_size
    }
    {
        sm_shared_by_blk = 0
        sm_shared_by_indx = 0
```

Mirror Specific Unit Structure (cont)

```
        sm_get_component_count = 0
        sm_get_bcss = 0
    }
    {
        sm_shared_by_blk = 0
        sm_shared_by_indx = 0
        sm_get_component_count = 0
        sm_get_bcss = 0
    }
    {
        sm_shared_by_blk = 0
        sm_shared_by_indx = 0
        sm_get_component_count = 0
        sm_get_bcss = 0
    }
```

Mirror Specific Unit Structure (cont)

```
]
un_mmic = {
        un_ovrlap_chn_mx = {
                _opaque = [0]
        }
        un_ovrlap_chn_cv = {
                _opaque = 0
        }
        un_ovrlap_chn = {
                dq = {
                        maxq_len = 0
                        qlen = 0
                        treqs = 0
                        dq_next = 0
```

/* overlap chain contains mirror parent save information for outstanding writes. Used to delay writes to same block until first write completes */

Mirror Specific Unit Structure (cont)

```
        dq_next = 0
        dq_prev = 0
        dq_call = 0
    }
    ps_bp = 0
    ps_un = 0
    ps_ui = 0
    ps_childbflags = 0
    ps_addr = 0
    ps_firstblk = 0
    ps_lastblk = 0
    ps_flags = 0
    ps_allfrom_sm = 0
    ps_writable_sm = 0
    ps_current_sm = 0
```

Mirror Specific Unit Structure (cont)

```
        ps_active_cnt = 0
        ps_frags = 0
        ps_changecnt = 0
        ps_ovrlap_next = 0
        ps_ovrlap_prev = 0
        ps_call = 0
        ps_mx = {
                _opaque = [0]
        }
}
un_resync_mx = {
        _opaque = [0]
}
un_resync_cv = {
        _opaque = 0
```

Mirror Specific Unit Structure (cont)

```
}
un_outstanding_writes = 0x60005622800          /* dirty region bitmap */
un_goingclean_bm = 0x60003898400
un_goingdirty_bm = 0x6005e99640
un_dirty_bm = 0x60005d56a38
un_resync_bm = 0x600018a0980
un_rs_buffer = 0
un_suspend_wr_flag = 0
un_suspend_wr_mx = {
        _opaque = [0]
}
un_suspend_wr_cv = {
        _opaque = 0
}
```

Mirror Specific Unit Structure (cont)

```
        un_suspend_wr_cv = {
                _opaque = 0
        }
    un_mirror_owner = 0
    un_resync_startbl = 0
    un_owner_mx = {
        _opaque = [0]
    }
    un_owner_state = 0
    un_mirror_owner_status = 0
    un_dmr_mx = {
        _opaque  = [0]
    }
```

Mirror Specific Unit Structure (cont)

```
        un_dmr_cv = {
                _opaque  = 0
        }
        un_dmr_last_read = 0
        un_rs_cprinfo = ;  (forward declaration)
        un_rs_cpr_mx = {
                _opaque = [0]
        }
        un_resync_completed = 0
        un_abr_count = 0
}
un_rrp_inflight_mx = {
        _opaque = [0]
}
```

Mirror Specific Unit Structure (cont)

```
un_rs_thread = 0                /* Multi-owner diskset resync */
un_rs_thread_mx = {
        _opaque = [0]
}
un_rs_thread_cv = {
        _opaque  = 0
}
un_rs_thread_flags = 0
un_rs_prev_ovrlap = 0
un_rs_resync_to_id = 0
un_rs_progress_mx  = {
        _opaque = [0]
}
un_rs_progress_cv = {
        _opaque = 0
}
```

Mirror Specific Unit Structure (cont)

```
    un_rs_progress_flags = 0
    un_rs_msg = 0
}
```

32 Bit OD Structures

- On disk structures for < 1TB metadevices
- Converted to corresponding structure incore
- Reverted back to 32 bit on disk structure when writing to mddb
  - EX. ms_comp32_od converted to ms_comp

Stripe

- md_sps – stripe parent save structure
- md_scs – stripe child save structure
- md_ms_unit – stripe unit structure
  - ms_row - stripe row structure (part of ms_unit)
  - ms_comp – stripe component structures (end of ms_unit)

Stripe Unit Structure

```
Typedef struct ms_unit {
        mdc_unit_t      c;
        int             un_hsp_id;      /* hot spare pool db record id */
        uint_t          un_nrows;       /* number of rows */
        uint_t          un_ocomp;       /* offset of ms_comp array */
        struct ms_row {
                int         un_icomp;       /* ms_comp array index of first comp */
                uint_t      un_ncomp;       /* # comps in this row */
                diskaddr_t  un_blocks;  /* total blocks in this row */
                diskaddr_t  un_cum_blocks;/* cumulative blocks in row */
                diskaddr_t  un_interlace;   /* # blks from each disk */
        } un_row [1];
} ms_unit_t;
```

ms_unit incore

metainit d20 2 1 c1t2d0s0 1 c1t3d0s0

```
> md_set::print md_set_t s_un[0]
      s_un[0] = 0x60002d8ca00
      > 0x60002d8ca00::print ms_unit_t
      {
            c = {
                  un_revision = 0
                  un_type = 1 (MD_DEVICE)        -> STRIPE
                  un_status = 0
                  un_parent_res = 0
                  un_child_res = 0
                  un_self_id = 0
```

ms_unit incore (cont)

un_record_id = 0x8
un_size = 0x138
un_flag = 0x1
un_total_blocks = 0x88f3800
un_actual_tb = 0x88f3800
un_nhead = 0x18
un_nsect = 0x1a8
un_rpm = 0x2729
un_wr_reinstruct = 0x11b
un_rd_reinstruct = 0
un_vtoc_id = 0
un_capabilities = 0xb
un_parent = 0xffffffff -> top level

ms_unit incore (cont)

```
        un_user_flags = 0
    }
    un_hsp_id = 0xffffffff
    un_nrows = 0x2              -> number of rows is 2
    un_ocomp = 0xa8             -> offset of comp array
    un_row = [
        {
            un_icomp = 0        -> index into comp arr is 0
            un_ncomp = 0x1
            un_blocks = 0x40980
            un_cum_blocks = 0x40980
            un_interlace = 0x400
        }
    ]
}
```

Stripe Rows

- Since there are 2 rows (un_nrows), then another row struct follows the unit struct.
- First row can be printed like this:

```
>0x60002d8ca00::print ms_unit_t un_row[0]
```

- 2nd row can be printed like this:

```
>0x60002d8ca00::print ms_unit_t un_row[1]
    mdb: index 1 is outside of array bounds [0...0]
    {
        un_row[1].un_icomp = 0x1    -> index to comp arr is 1
        un_row[1].un_ncomp = 0x1
        un_row[1].un_blocks = 0x88b2e80
        un_row[1].un_cum_blocks = 0x88f3800
        un_row[1].un_interlace = 0x400
    }
```

Stripe Component Array

```
>0x60002d8ca00+a8::print ms_comp_t {      /* starts 0xa8 from begin of struct */
        un_key = 0x3
        un_dev = 0x7600000018             /* major = 0x76 -> ssd, minor = 0x18 */
        un_start_block = 0
        un_mirror = {                     /* mirror shared data (md_m_shared) data
            ms_flags = 0                     used by mirror subdriver but in stripe
            ms_state = 0x1                   comp structure */
            ms_lasterrcnt = 0
            ms_orig_dev = 0
            ms_orig_blk = 0
            ms_hs_key = 0
            ms_hs_id = 0
            ms_timestamp = {
                tv_sec = 0x43cd76a9
                tv_usec  = 0xdc243
            }
corresponds to comp: 1 root    sys    118, 24   Jan 17  15:58  /dev/rdsk/c1t2d0s0
```

Stripe Component Array - 2nd comp

```
>0x60002d8ca00+a8+(1*48)::print ms_comp_t
        {
                un_key = 0x4
                un_dev = 0x7600000000           /* major 118, minor 0 */
                un_start_block = 0
                un_mirror = {
                        ms_flags  = 0
                        ms_state = 0x1
                        ms_lasterrcnt = 0
                        ms_orig_dev = 0
                        ms_orig_blk = 0
                        ms_hs_key = 0
                        ms_hs_id = 0
                        ms_timestamp = {
                                tv_sec  = 0x43cd76a9
                                tv_usec = 0xdc243
                        }
```

48 = sizeof(ms_comp_t)

Corresponds to comp: 1 root sys 118, 0 Jan 17 15:58 /dev/rdsk/c1t3d0s0

Raid

- md_raidps - raid parent save structure
- md_raidcs – raid child save structure
- md_raidcbuf
- mr_column
- mr_column_ic – incore only column info
- mr_unit – raid unit structure
- mr_unit_ic – incore only raid unit info has array of [1] column at end
- raid_pwhdr – pre write area

mr_unit

```
typedef struct mr_unit {
        mdc_unit_t      c;
        int             un_raid_res;
        uint_t          un_magic;
        rus_state_t     un_state;
        md_timeval32_t  un_timestamp;   /* 32 bit fixed size */
        uint_t          un_origcolumncnt;
        uint_t          un_totalcolumncnt;
        uint_t          un_rflags;
        uint_t          un_segsize;
        diskaddr_t      un_segsincolumn;
        uint_t          un_maxio;       /* in blks */
        uint_t          un_iosize;      /* in blks */
        uint_t          un_linlck_flg;
```

mr_unit (cont)

```
 uint_t          un_pwcnt;
uint_t          un_pwsize;
long long       un_pwid;
uint_t          un_percent_done;
uint_t          un_resync_copysize;     /* in blks */
hsp_t           un_hsp_id;
/*
 * This union has to begin at an 8 byte aligned address.
 * If not, this structure has different sizes in 32 / 64 bit
 * environments, since in a 64 bit environment the compiler
 * adds paddings before a long long, if it doesn't start at an 8byte
 * aligned address.
 * Be careful if you add or remove structure elements before it!
 */
```

mr_unit (cont)

```
union   {
        struct  {
                diskaddr_t      _t_un_resync_line_index;
                uint_t          _t_un_resync_segment;
                int             _t_un_resync_index;
        } _resync;
        struct  {
                diskaddr_t      _t_un_grow_tb;
                uint_t          _t_un_init_colcnt;
                u_longlong_t    _t_un_init_iocnt;
        } _init;
} _t_un;
```

mr_unit (cont)

```
        union {
            mr_unit_ic_t    *_mr_ic;
            uint_t          _mr_ic_pad[2];
        } un_mr_ic;

        mr_column_t     un_column[1];
} mr_unit_t;
```

Softpart

- md_spps – softpart parent save structure
- md_spcs – softpart child save structure
- mp_watermark – watermark structure
- mp_unit – soft partition unit structure array of [1] at end of mp_unit

mp_unit

```
struct mp_unit  {
                mdc_unit_t      c;
                mdkey_t         un_key;
                md_dev64_t      un_dev;
                sp_ext_offset_t       un_start_blk;
                sp_status_t     un_status;
                uint_t          un_numexts;
                sp_ext_length_t       un_length;
                mp_ext_t        un_ext[1];
};
```

Mirror

- md_mps – mirror parent save structure
- md_mcs – mirror child save structure
- mm_unit – mirror unit structure (shown earlier)
- md_m_shared – used by mirror but in stripe struct

Write on Write

- Used with memory mapped I/O page to see if page was dirtied while write was in progress.
- If dirty issue another write
- Disabled by default since kills performance

Overlapping Writes

- Check that I/O request does not cause overlap with an already pending I/O.
- If it does, block until the overlapped I/O completes

Solaris Volume Manager : Kernel Data Structures