

# IOCTL/new ABI Status

Jason Gunthorpe  
Linux Plumbers 2018



# IOCTL Status

- IOCTL Infrastructure is now the only way to access some new functionality
  - Device Memory
  - Flow Actions
  - mlx5 driver functions (devx, flow, etc)
- Modifications to structs for write() are now forbidden
  - New user API functionality must go to IOCTL
  - Reformat existing write() into ioctl() as required

# API Improvements

- **Attribute Language updates:**
  - Object array of IDR numbers
  - Sub-type tagged pointer (enum)
  - Constant value from an enum of choices
  - Bitwise Flags
- **Internal Structure**
  - Use radix tree library instead of open coding
  - Simpler #define macros
  - Revised uobject handling and locking
  - List based spec language instead of tree based (forthcoming)

# API Improvements

- More refined definition language:

```
DECLARE_UVERBS_NAMED_METHOD(  
    UVERBS_METHOD_DM_ALLOC,  
    UVERBS_ATTR_IDR(UVERBS_ATTR_ALLOC_DM_HANDLE,  
                    UVERBS_OBJECT_DM,  
                    UVERBS_ACCESS_NEW,  
                    UA_MANDATORY),  
    UVERBS_ATTR_PTR_IN(UVERBS_ATTR_ALLOC_DM_LENGTH,  
                        UVERBS_ATTR_TYPE(u64),  
                        UA_MANDATORY),  
    UVERBS_ATTR_PTR_IN(UVERBS_ATTR_ALLOC_DM_ALIGNMENT,  
                        UVERBS_ATTR_TYPE(u32),  
                        UA_MANDATORY));  
DECLARE_UVERBS_NAMED_OBJECT(UVERBS_OBJECT_DM,  
                             UVERBS_TYPE_ALLOC_IDR(uverbs_free_dm),  
                             &UVERBS_METHOD(UVERBS_METHOD_DM_ALLOC));  
const struct uapi_definition uverbs_def_obj_dm[] = {  
    UAPI_DEF_CHAIN_OBJ_TREE_NAMED(UVERBS_OBJECT_DM,  
    UAPI_DEF_OBJ_NEEDS_FN(dealloc_dm)),
```

# API Improvements

- Consolidation on 'struct uverbs\_attr\_bundle' (forthcoming)
  - All method handlers write/write\_ex/ioctl have the same call-in signature:  
`static int handler(struct uverbs_attr_bundle *attrs)`
- General allocator for handler calls:
  - `void *uverbs_alloc(struct uverbs_attr_bundle *bundle, size_t size)`
  - Always frees the memory when the handler exits
  - Small amount of stack memory available to this allocator

# Fork Support (forthcoming)

- Give up on converting all APIs to native IOCTL
  - Provide an ioctl() function that can invoke 'write' or 'write\_ex' handlers using the same ABI as write()
  - Very hard as all existing write handlers make assumptions about user memory layouts – have to remove all assumptions first

```
DECLARE_UVERBS_NAMED_METHOD(UVERBS_METHOD_INVOKE_WRITE,  
                             UVERBS_ATTR_CONST_IN(UVERBS_ATTR_WRITE_CMD,  
                                                    enum ib_uverbs_write_cmds,  
                                                    UA_MANDATORY),  
                             UVERBS_ATTR_PTR_IN(UVERBS_ATTR_CORE_IN,  
                                                  UVERBS_ATTR_MIN_SIZE(sizeof(u32)),  
                                                  UA_OPTIONAL),  
                             UVERBS_ATTR_PTR_OUT(UVERBS_ATTR_CORE_OUT,  
                                                  UVERBS_ATTR_MIN_SIZE(0),  
                                                  UA_OPTIONAL),  
                             UVERBS_ATTR_UHW());
```

# Enable IOCTL by default

- IOCTL CQ should probably be revised, looks strange now
- Remove INFINIBAND\_EXP\_LEGACY\_VERBS\_NEW\_UAPI
- Switch rdma-core to IOCTL\_MODE=both by default
  
- When?
  
- strace support?

# Fix fork support in RDMA-CM

- Rework in same IOCTL scheme?
- Just add a new wrapper IOCTL like write?
- Something else?