www.storagedeveloper.org

Use only what you really need – Kernel space buffers in server process

Rafal Szczesniak
DellEMC / Isilon

To copy or not to copy?

Available alternatives

Introducing the io vector

Utilising the io vector

Use cases

The future

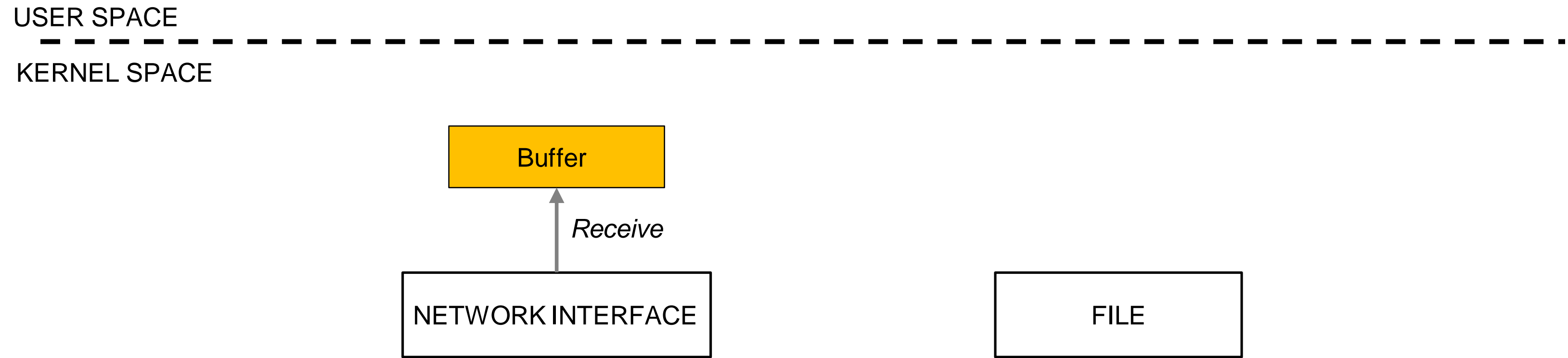
2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Typical communication with the network

- Arriving packets are held in the kernel buffers
- User process reading from the bsd socket - **copyout**
- Writing the data to a file (descriptor) – **copyin**
- Lots of CPU cycles just to copy the data back and forth

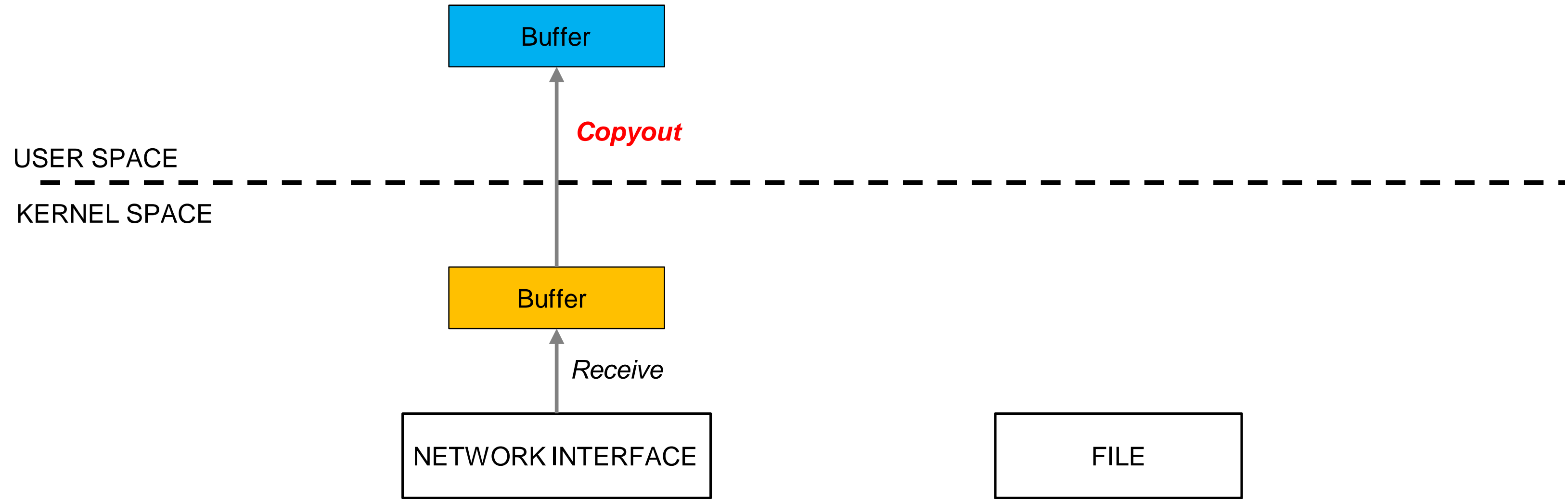
2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Typical communication with the network



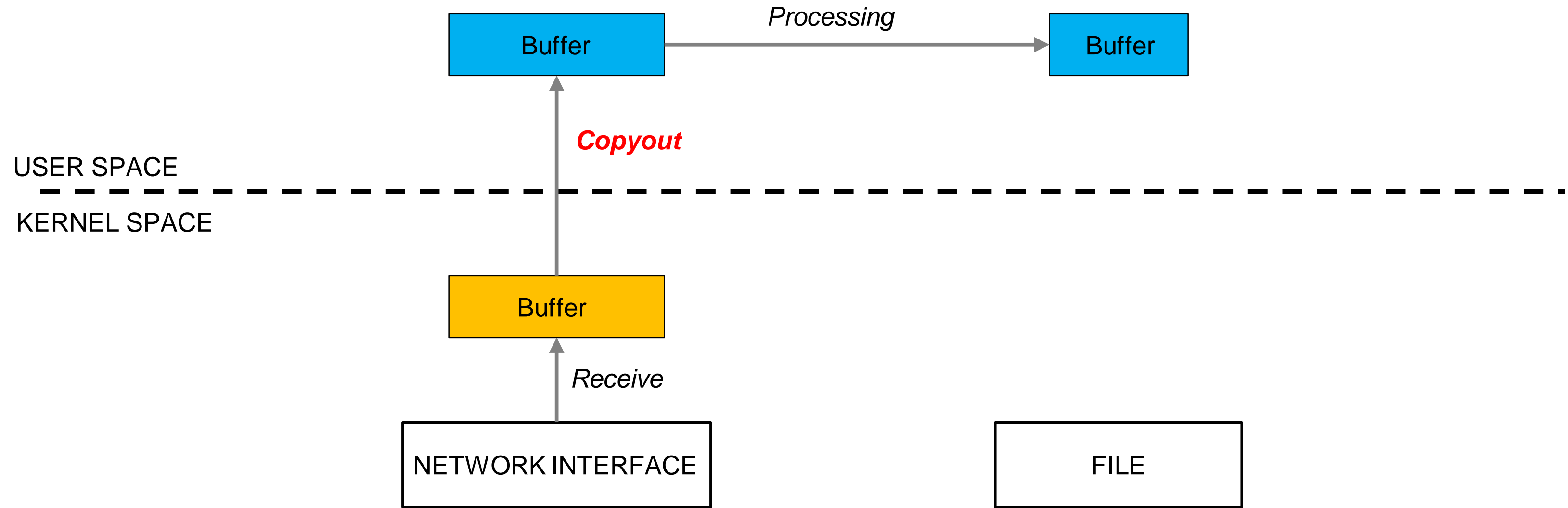
2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Typical communication with the network



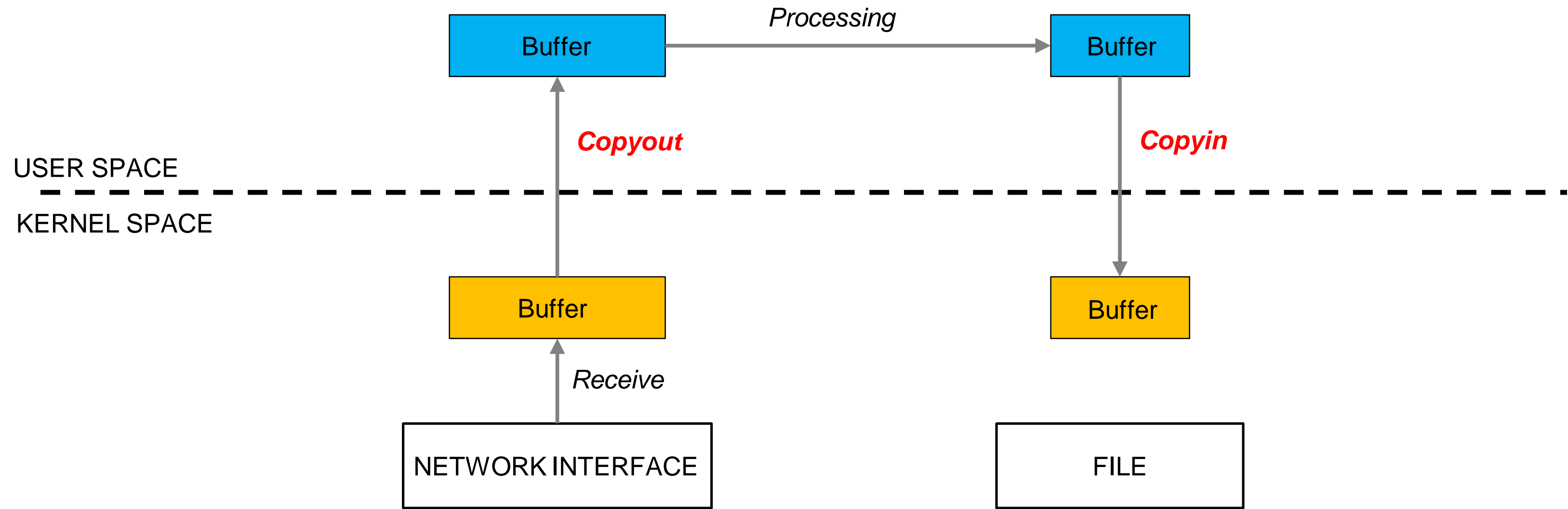
2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Typical communication with the network



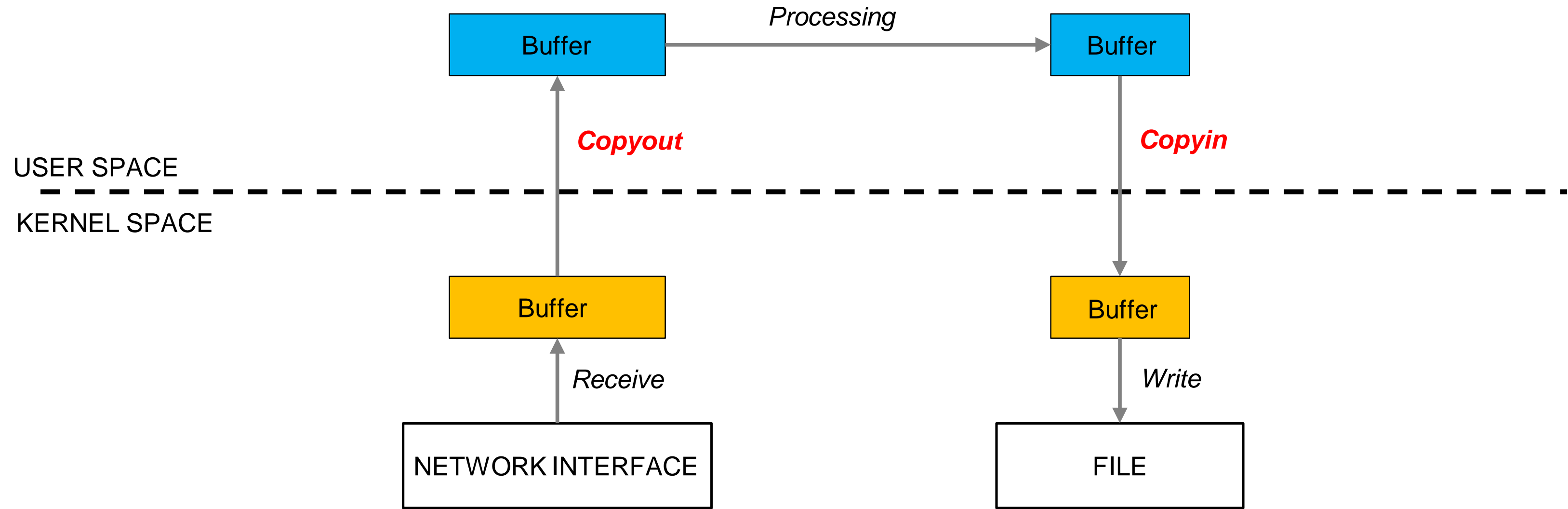
2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Typical communication with the network



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Typical communication with the network



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Networks are faster and more reliable

- The larger the MTU (Maximum Transmission Unit) the more data to copy
- Significant percentage of the CPU time can be spent on copying alone

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

To copy or not to copy?

Available alternatives

Introducing the io vector

Utilising the io vector

Use cases

The future

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Zero-copy

- An attempt to avoid wasting the CPU cycles
- Not a new concept
- Typically utilising file descriptors for accessing the data
- Various implementations among various OSes

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Zero-copy and splice(2)

- One side of the transfer has to be a pipe fd
- This is mostly due to what pipe buffers support
- Two extra file descriptors per transfer
- Linux-specific syscall

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Zero-copy and sendfile(2)

- Input file descriptor must allow mmap (typically a file)
- Obviously, it cannot receive

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

To copy or not to copy?

Available alternatives

Introducing the io vector

Utilising the io vector

Use cases

The future

SDC 18

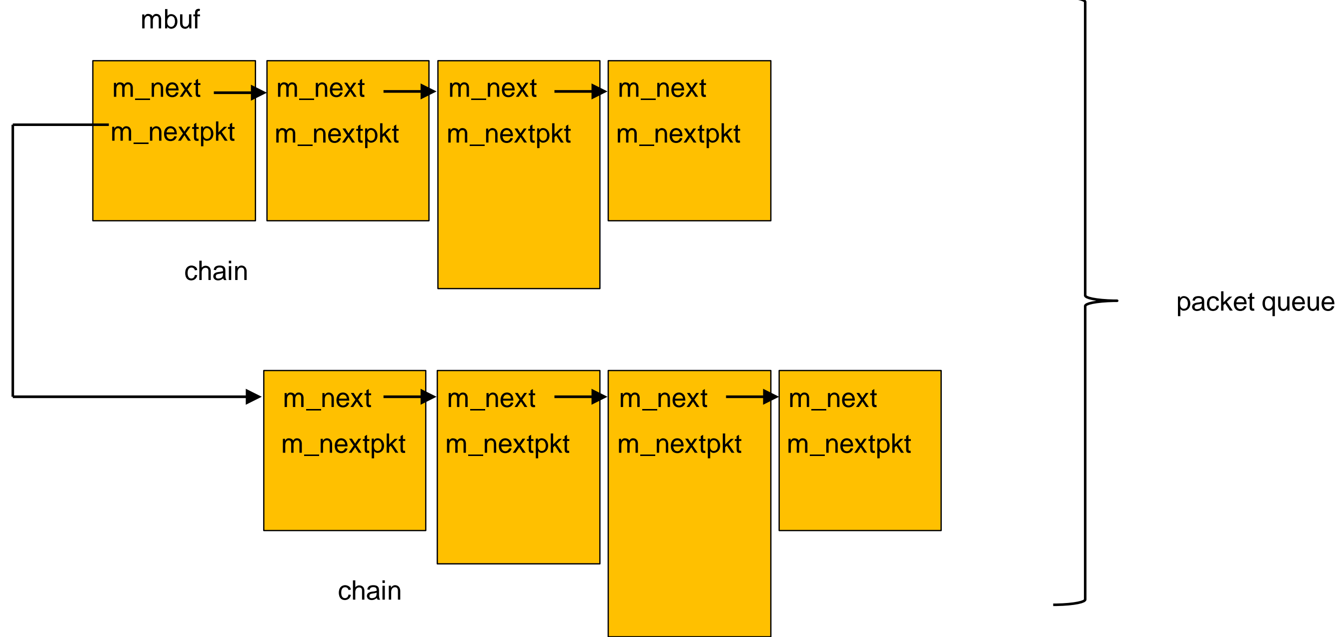
2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Engaging a kernel-level structure - mbuf

- Simple structure holding a buffer pointer in FreeBSD
- Primarily used in IPC, so it facilitates certain network communication features
- mbufs can be chained together in a list (mbuf chain)
- Chains and also be linked (packet queue)

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

mbuf chains



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Entering user space – io vector

- Follows the same scatter-gather pattern utilising a list of entries
- Gets translated to an mbuf chain when in the kernel space
- Not everything is needed in the user space – different buffer types can be treated differently

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

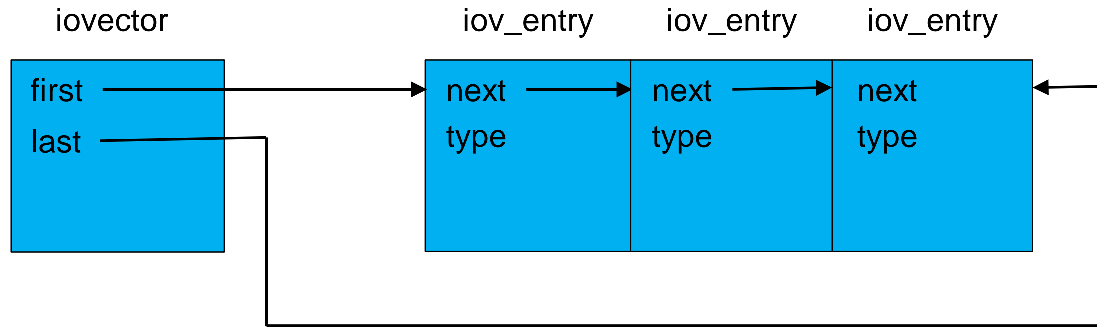
Entering user space – io vector (contd.)

```
struct iov_entry {
    struct iov_entry *next;
    enum iov_entry_type type;
    size_t size;
    size_t length;
};

struct iovector {
    struct iov_entry *first;
    struct iov_entry *last;
};
```

SDC 18

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

IO Vector



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Memory entry

- Represents a classic memory buffer (void*, size)
- The buffer is allocated in the user space, so it can be modified
- Passing it down to the kernel is going to make its copy (in mbufs)

SDC 18

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Memory entry (contd.)

```
struct iov_entry_memory {
    struct iov_entry *next;
    enum iov_entry_type type;       /* = memory */
    size_t size;
    size_t length;
    char data[];
};
```


2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Memory entry (contd.)

struct iov_entry_memory

next
type = memory
size
length
data

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Kernel entry

- Represents a buffer living in the kernel space
- Not accessible from the user space directly
- Passed down to the kernel space does not need a copy

SDC 18

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Kernel entry

- Represents a buffer living in the kernel space
- Not accessible from the user space directly
- Passed down to the kernel space does not need a copy
- **Accessible through a file descriptor**

SDC 18

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Kernel entry (contd.)

```
struct iov_entry_kernel {
    struct iov_entry *next;
    enum iov_entry_type type; /* = kernel */
    size_t size;
    size_t length;
    int fd;
};
```


2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Kernel entry (contd.)

struct iov_entry_kernel

next
type = kernel
size
length
fd

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

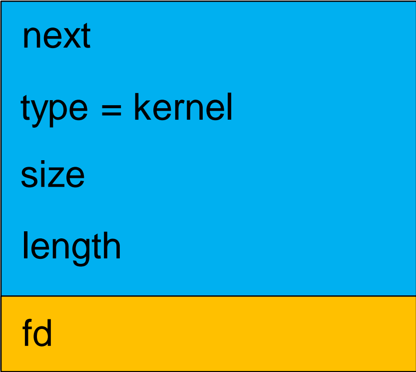
Kernel entry (contd.)

- A file descriptor needs its struct file* in the kernel
- An mbuf chain can be treated as a pseudo-file and support the basic operations
- At the user level it is just passed around
- At the kernel level it can be used for:
  - Reading an writing (file)
  - Receiving and sending (socket)

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

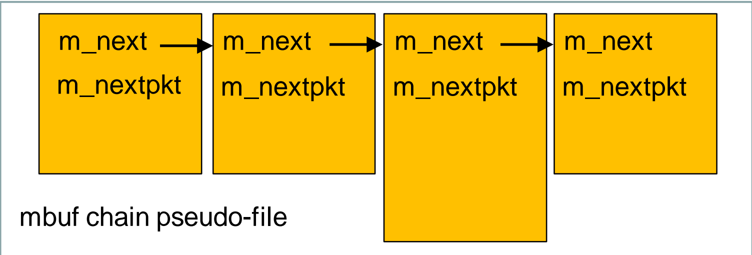
Kernel entry (contd.)

struct iov_entry_kernel



USER SPACE

KERNEL SPACE



struct file*

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

To copy or not to copy?

Available alternatives

Introducing the io vector

Utilising the io vector

Use cases

The future

SDC 18

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Flexibility through chaining

- Both entry types can represent any blob of data:
  - Packet header allocated in the user space
  - Contents read from a file
- Building a packet is much easier (header, request/response, padding can all be separate)

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Flexibility through chaining (contd.)

```
int error;
struct iovector vec;
struct iov_entry_memory hdr, res;
struct iov_entry_kernel data;

error = IovInit(&vec);
CLEANUP_ON_ERROR(error);

error = IovCreateMemoryEntry(&hdr, /* size */);
CLEANUP_ON_ERROR(error);

/* Prepare the header */

error = IovPushBack(&vec, (struct iov_entry*)&hdr);
CLEANUP_ON_ERROR(error);
```

SDC 18

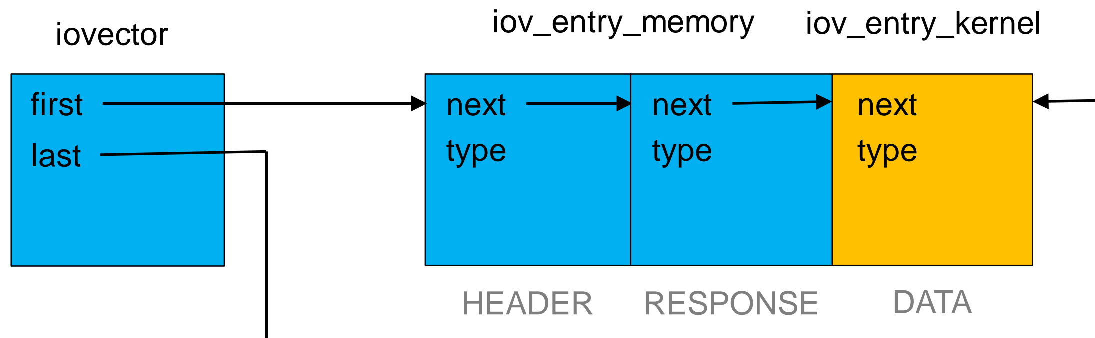
2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Flexibility through chaining (contd.)

```
error = IovCreateMemoryEntry(&res, /* size */);
CLEANUP_ON_ERROR(error);

/* Prepare the response */

error = IovPushBack(&vec, (struct iov_entry*)&hdr);
CLEANUP_ON_ERROR(error);

error = IovCreateKernelEntry(&data, /* size */);
CLEANUP_ON_ERROR(error);

/* Read the data */

error = IovPushBack(&vec, (struct iov_entry*)&data);
CLEANUP_ON_ERROR(error);
```


2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Flexibility through chaining (contd.)



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Flexibility through splitting and converting

- An io vector can have its parts split off and freed when no longer needed
- Kernel entries can be pulled up to the user space
- Memory entries can be pushed down, too
- New syscalls are required

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Flexibility through splitting and converting

```
int error;
struct iovector received, hdr_vec, req_vec, data;
struct smb2_header *header;

/* Vector received – now, get the header */

error = IovPullup(&received, sizeof(*header));
CLEANUP_ON_ERROR(error);

error = IovSplit(&receive, sizeof(*header), &hdr_vec);
CLEANUP_ON_ERROR(error);

header = IovMemOrigin(&hdr_vec);

/* Process the header */
```

SDC 18

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Replacing a buffer with an io vector

- send/recv/read/write could take an io vector instead of a buffer
- This enables keeping a part or all of the data read/received in the kernel
- We can always pull up when needed

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

To copy or not to copy?

Available alternatives

Introducing the io vector

Utilising the io vector

Use cases

The future

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

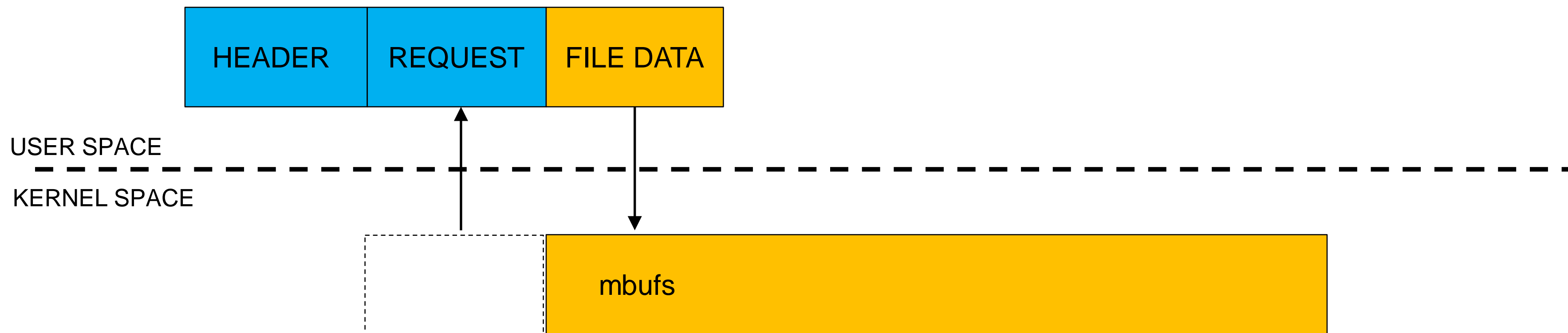
Use case – large SMB2 write (1/5)

- A packet is received in a minimum-sized iovector
- The header says it's a write request

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – large SMB2 write (2/5)

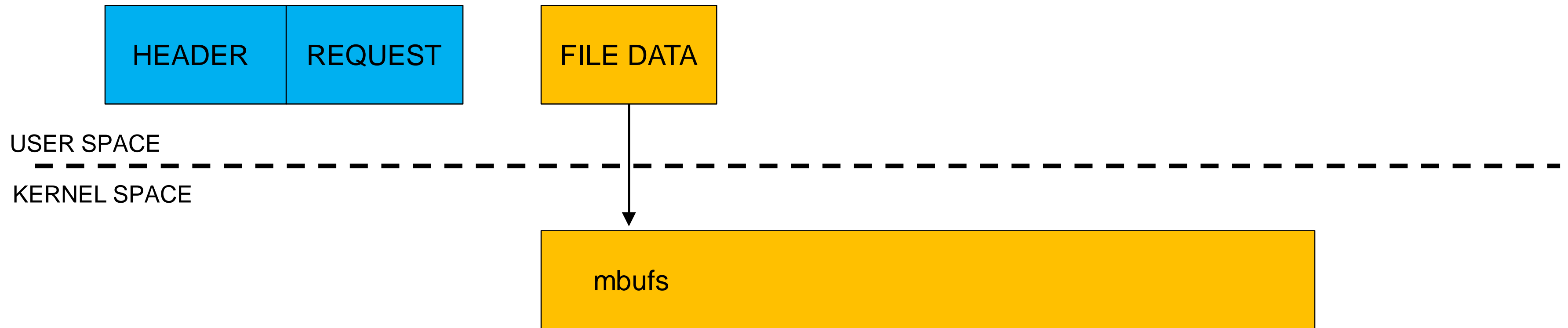
- Pull up the request (fixed-size)



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – large SMB2 write (3/5)

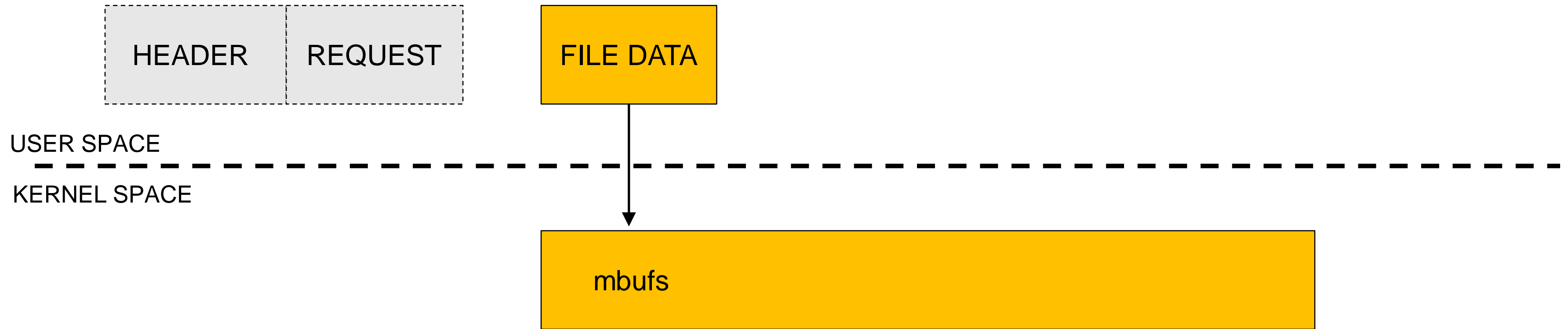
- Split off the SMB2 packet to a separate iovector
- File data is separate now, too



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – large SMB2 write (4/5)

- Free what's no longer needed
- Write to the file – mbufs will be moved in-kernel



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

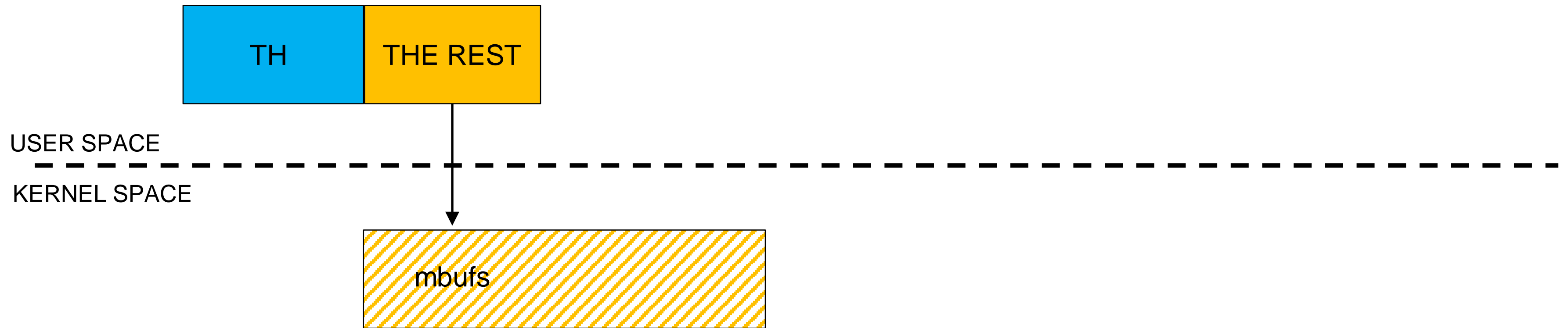
Use case – large SMB2 write (5/5)

- The data written to the file has never left the kernel
- The packet can be received to the “minimum necessary” iovector (only the SMB2 header in user space)
- What’s been processed and no longer needed can be released early

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – SMB3 decryption (1/6)

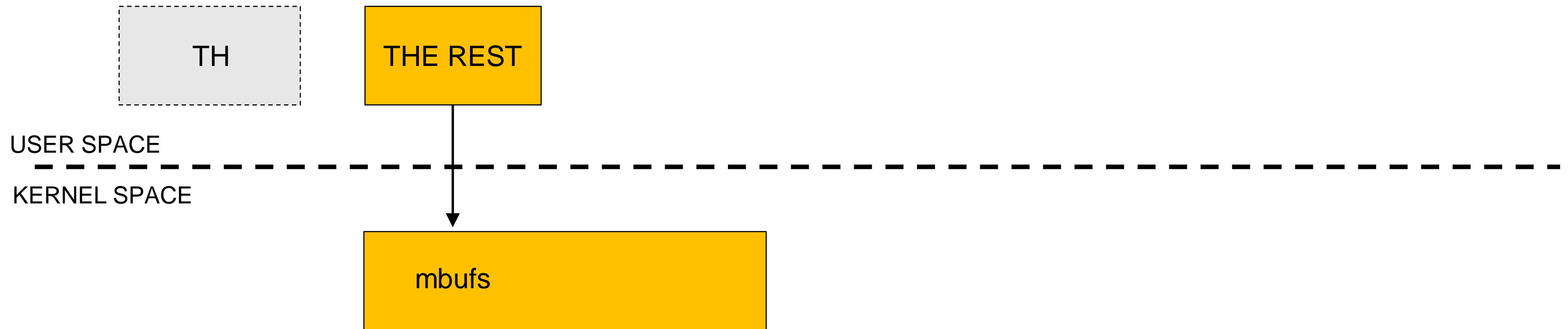
- A packet is received in a minimum-sized iovector
- The header is a TRANSFORM_HEADER



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – SMB3 decryption (2/6)

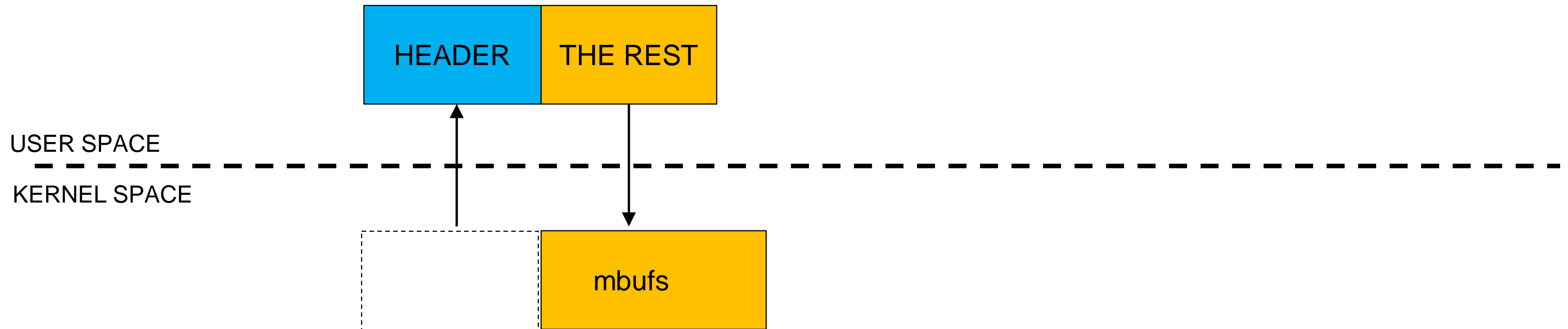
- Decrypt the rest first
- Split off and free TRANSFORM_HEADER



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – SMB3 decryption (3/6)

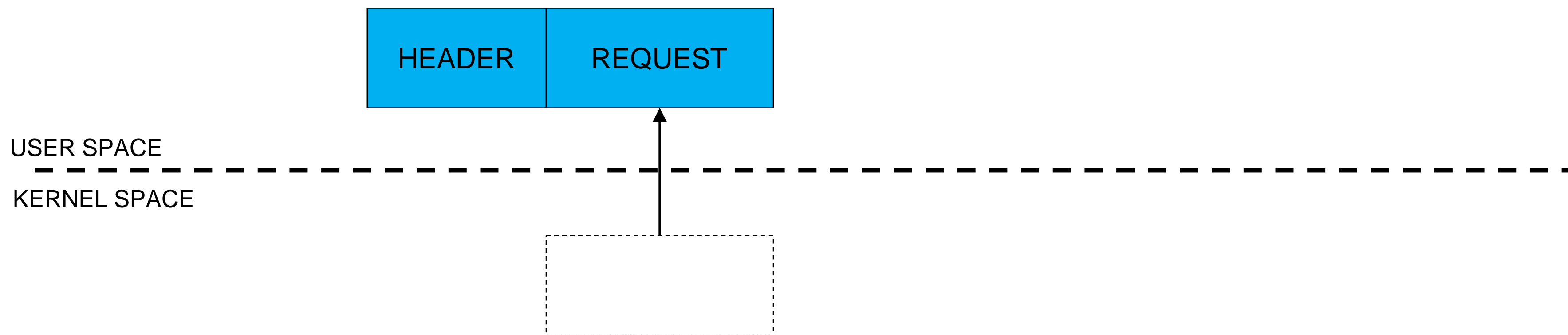
- Pull up SMB2 header (it’s decrypted now)
- The header says it’s a change notify request



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – SMB3 decryption (4/6)

- Pull up the request



2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – SMB3 decryption (5/6)

- Execute and free the packet

USER SPACE

KERNEL SPACE

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Use case – SMB3 decryption (6/6)

- Decryption can be done both over the user and kernel space buffers
- Sometimes we end up pulling up everything

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

To copy or not to copy?

Available alternatives

Introducing the io vector

Utilising the io vector

Use cases

The future

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

This is today, what's tomorrow?

- How difficult would it be with the Linux kernel?
- RDMA and the like

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Linux kernel

- There is no mbuf in Linux, some sources say sk_buff is the equivalent
- sk_buff is **much** bigger and not necessarily useful here
- iov_iter appears to fit the bill much better
- BSD-style UIO is in place
- It would still require new syscalls

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

RDMA

- If an mbuf chain can be a pseudo-file, then any data coming out of an RDMA-enabled device could be, too
- Functions implementing the file ops would have to be different
- Another iovector entry type would be needed
- Perhaps something for a recommendation?

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.

Thank you!

Questions?

rafal.szczesniak@dell.com
@emc.com

Credits: Brian Koropoff
(for laying the groundwork of the kernel code)

2018 Storage Developer Conference. © DellEMC / Isilon. All Rights Reserved.