Indirect Memory registration

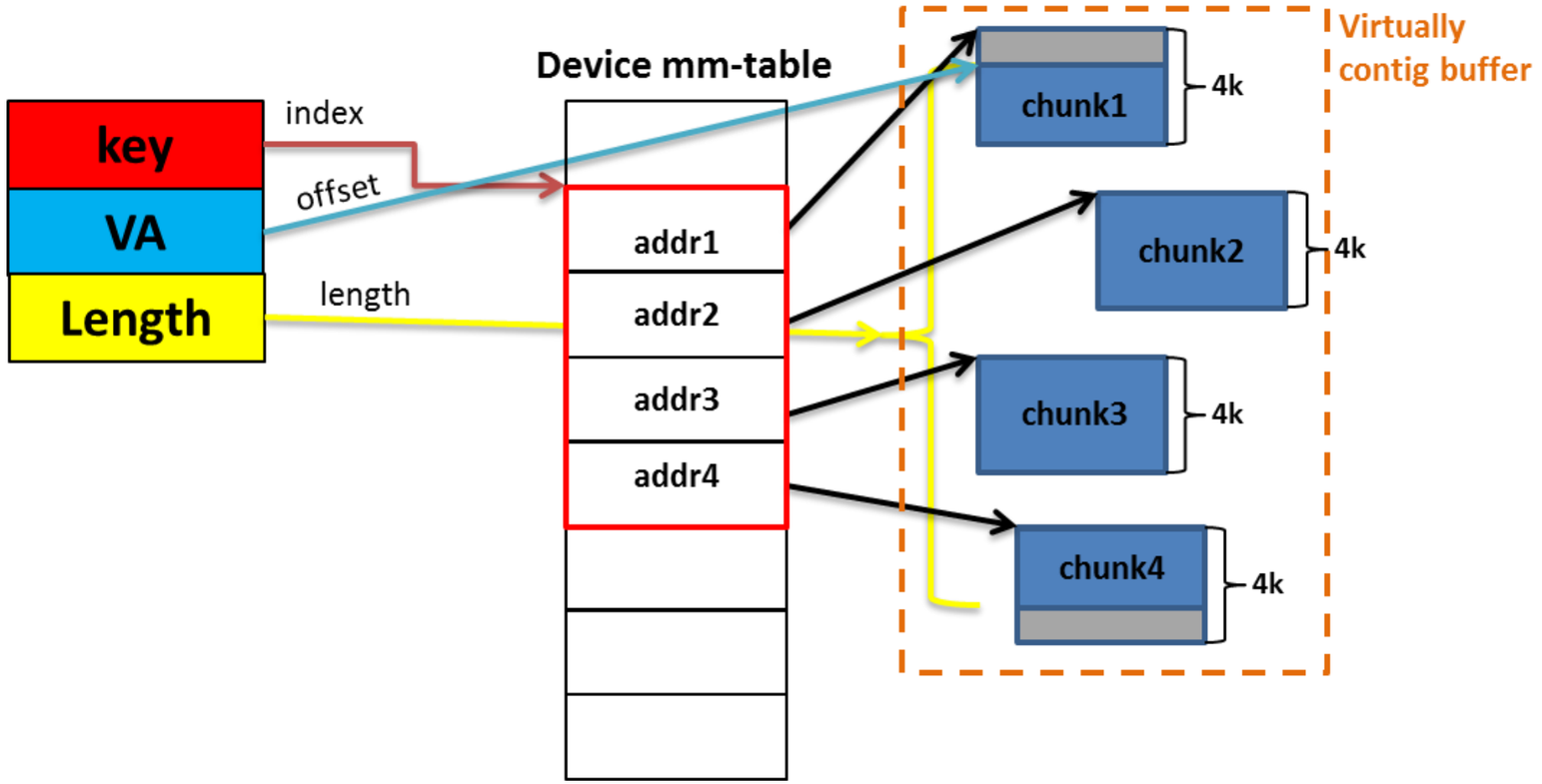
Sagi Grimberg

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™

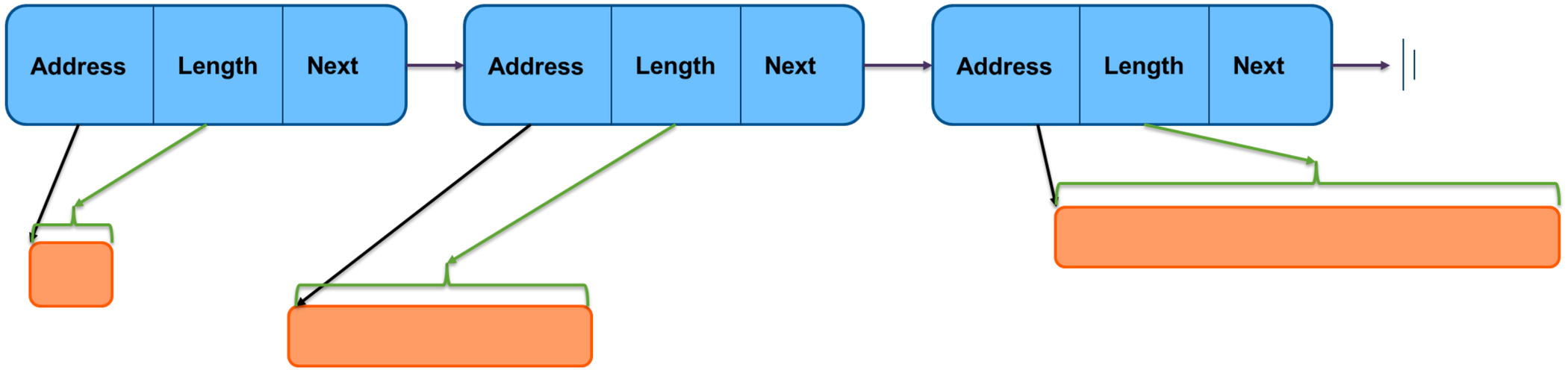
Memory Registration Today

- RDMA memory registration API allows a specific memory layout (scatter-gather list of buffers)
  - All buffers must be block aligned (single page_shift)
  - The first buffer is allowed to have a FBO (First Byte Offset)
  - The last buffer is allowed to end before the EOB (End Of Block)



Problem statement

- How can we register an arbitrary SG list of buffers?



- Possible Solutions:
  - Use multiple MRs to register a single SG list (each MR will register an “aligned segment”)
  - Use Bounce buffers (SW allocates a well-aligned buffer and copies)
  - Break SG list across multiple transactions.
  - Use a block size that satisfies the minimal block alignment.

Use cases

- Block/file Storage:
  - Vectored Direct IO
  - SG_IO
  - Block layer bio merges
  - T10-PI (much easier to get out of alignment in 8-byte DIF buffers merges)

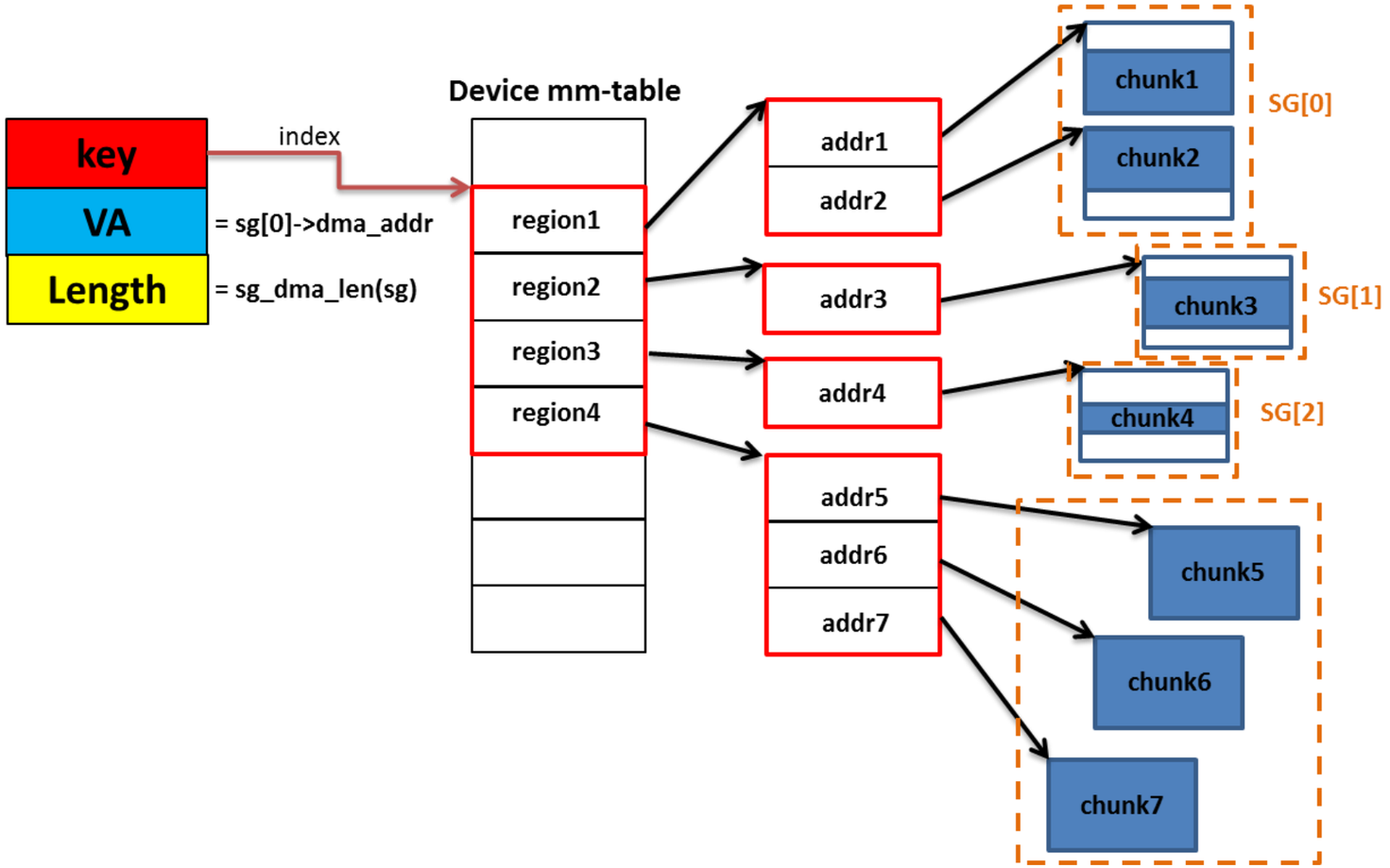
- MPI:
  - Single RDMA to/from a Remote scatter (in user-space)
  - “User-space FRWR”

- PMEM devices – byte addressable storage (NVM programming model)
  - We can no longer afford to work-around this limitation and meet the latency requirements.
    - No bounce buffering what-so-ever.
    - Efficient multiple byte range remote access.
  - See http://www.snia.org/sites/default/files/DougVoigt_RDMA_Requirements_for_HA.pdf

- More?

Indirect Memory registration

- Enhance the RDMA stack to allow registering an arbitrary SG lists for devices that support it.



API

- Follow the same notation as standard FRWR.
  - Instead of passing a page vector to the device we pass a vector of ib_sge elements.

OP	FRWR	Indirect

Allocate	`frpl = ib_alloc_fast_reg_page_list()`
	
	`mr = ib_alloc_fast_reg_mr()`
		`irl = ib_alloc_indir_reg_list()`
		`mr_attrs.create_flags = IB_MR_INDIRECT_REG;`
		`mr = ib_create_mr(mr_attrs)`

Free	`ib_free_fast_reg_page_list(frpl)` `ib_dereg_mr(mr)`	`ib_free_indir_reg_list(irl)` `ib_dereg_mr(mr)`

Register interface (post_send)	`struct {`
	`  u64 iova_start;`
	`  struct ib_fast_reg_page_list *page_list;`
	`  unsigned int page_shift;`
	`  unsigned int page_list_len;`
	`  u32 length;`
	`  int access_flags;`
	`  u32 rkey;`
	`} fast_reg;`
		`struct {`
		`  u64 iova_start;`
		`  struct ib_indir_reg_list *indir_list;`
		
		`  unsigned int indir_list_len;`
		`  u64 length;`
		`  unsigned int access_flags;`
		`  u32 mkey;`
		`} indir_reg;`

Opcode	`IB_WR_FAST_REG_MR`	`IB_WR_REG_INDIR_MR`

Discussion

- Method for generic SG support
  - Synchronous API (e.g. reg_mr)
  - Asynchronous API (e.g. frwr)
- Comments/Feedback on the API itself
  - Combine with FRWR API?
- Requirement and place for abstraction

Thank You

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™