Signature Verbs Extension

Richard L. Graham
Data Integrity Field (DIF)

- Used to provide data block integrity check capabilities (CRC) for block storage (SCSI)
- Proposed by the T10 committee
- DIX extends the support to main memory
- Motivation:
  - Data integrity checks are included in:
    - memory bus
    - I/O bus (PCI-e)
    - internal chips
    - RAID controller within arrays
    - network packet interfaces
  - Missing data protection: storage controller
    - iSCSI can DMA wrong pages from memory and calculate checksum on these pages
    - Array may receive incorrect data and use it to generate RAID parity blocks
    - Disks may write to the wrong logical block
What is DIF

• The standard specify an additional 8 byte field designated for data integrity/protection for each data block (usually of size 512 bytes but not a must).

• GUARD tag (Logical Block Guarding):
  – 16-bit CRC covering the hardware sector
  – Regardless of sector size
  – 4096 KB sectors appear only to gain momentum in lower end

• REFERENCE tag (Misdirected writes):
  – 4 bytes – depend on protection type
  – For Type 1 protection, REF tag contains lower 32 bits of LBA
  – For Type 2 protection, REF tag has to match LBA in CDB + N
  – Wraps at 2TB with 512 byte sectors, 16TB with 4KB

• APPLICATION tag (Up for grabs):
  – 2 bytes per sector
  – Ownership negotiated with target
System Architecture

- Host Machine
- Memory
- Legacy Initiator
- Host Machine
- Memory
- Initiator with *DIF Offload Support
- Target Machine
- Memory
- DIF Enabled Target *DIF Offload Support
- Storage
- FC/SATA/SCSI

Data
Meta Data (DIF)
System Architecture

IO Read

[Diagram showing the flow of data between memory and storage, with labels for send, RDMA write, validate, H/W Ack, and send status.]
System Architecture
1. Allocate Signature enabled memory regions
   mr_init_attr.flags |= IB_MR_SIGNATURE_EN;
   sig_mr = ib_create_mr(pd, &mr_init_attr);

2. Set QP as Signature enabled
   qp_init_attr.create_flags |= IB_QP_CREATE_SIGNATURE_EN;
   sig_qp = ib_create_qp(pd, &qp_init_attr);

3. Register Signature MR (send work request IB_WR_REG_SIG_MR)
   sig_wr.opcode = IB_WR_REG_SIG_MR;
   sig_wr.sg_list = data_sge;            /* Data buffer */
   sig_wr.wr.sig_handover.prot = prot_sge;   /* protection buffer */
   sig_wr.wr.sig_handover.sig_attrs = &sig_attrs; /* signature attributes struct */
   sig_wr.wr.sig_handover.sig_mr = pi_ctx->sig_mr; /* Signature enabled MR */
   ret = ib_post_send(qp, sig_wr, &bad_wr);

3.5. do RDMA (data-transfer) – Leverage existing verbs support

4. Check Signature status
   ret = ib_check_mr_status(sig_mr, IB_MR_CHECK_SIG_STATUS, &mr_status);
Leverage Extended User Mode Memory Registration

- Memory Key Creation: Support combining contiguous registered memory regions into a single memory region. H/W treats them as a single contiguous region (and handles the non-contiguous regions)

- For a given memory region, supports non-contiguous access to memory, using a regular structure representation – base pointer, element length, stride, repeat count.
  - Can combine these from multiple different memory keys

- Memory descriptors are created by posting WQE’s to fill in the memory key

- Supports local and remote non-contiguous memory access
  - Eliminates the need for some memory copies
Combining Contiguous Memory Regions

3 memory regions
Each referenced by a different memory key

Registered Memory Region

v0
v1
v2
v3
v4
v5

One memory region
Referenced by one memory key
Non-contiguous in virtual memory

v0-v1
v2-v3
v4-v5
Non-Contiguous Memory Access – Regular Access

Contiguous Memory Addresses

Memory

Wire
Non-Contiguous Memory Access – Regular Access
Example in SCSI transport

**Setup MKey for signature operation**
- Input: data_sge, prot_sge, signature_mr
- Source Domain – wire
- Target Domain – memory
- Need small fence

**Send SCSI READ request which includes the signature_mr rkey**

**Setup MKey for signature operation**
- Input: data_sge, prot_sge, signature_mr
- Source Domain – memory
- Target Domain – wire
- Need small fence

**Post RDMA write**
- rkey = signature_mr rkey
- rkey = Received rkey

**Poll for write completion**

**Check on Signature outcome On signature_mr**

**Send outcome of Signature operations**

**Invalidate Signature MKey**
- MKey=signature_mr MKey

**Report results back to SCSI layer**

**Poll for receive completion**

**Check on Signature outcome On signature_mr**

**Invalidate Signature MKey**
- MKey=signature_mr MKey

**Send outcome of Signature operations**

**Invalidate Signature MKey**
- MKey=signature_mr MKey

**Post RDMA/Read**
- rkey = signature_mr rkey
- rkey = Received rkey

**Poll for write completion**

**Check on Signature outcome On signature_mr**

**Send outcome of Signature operations**

**Invalidate Signature MKey**
- MKey=signature_mr MKey
Thank You

#OFADevWorkshop