NAME

ibv_post_send − post a list of work requests (WRs) to a send queue

SYNOPSIS

#include <infiniband/verbs.h>

int ibv_post_send(struct ibv_qp **qp***, struct ibv_send_wr** **wr***,**
struct ibv_send_wr ***bad_wr***);**

DESCRIPTION

ibv_post_send() posts the linked list of work requests (WRs) starting with *wr* to the send queue of the queue pair *qp*. It stops processing WRs from this list at the first failure (that can be detected immediately while requests are being posted), and returns this failing WR through *bad_wr*.

The argument *wr* is an ibv_send_wr struct, as defined in <infiniband/verbs.h>.

```
struct ibv_send_wr {
        uint64_t              wr_id;              /* User defined WR ID */
        struct ibv_send_wr    *next;              /* Pointer to next WR in list, NULL if last WR */
        struct ibv_sge        *sg_list;           /* Pointer to the s/g array */
        int                   num_sge;            /* Size of the s/g array */
        enum ibv_wr_opcode    opcode;             /* Operation type */
        int                   send_flags;         /* Flags of the WR properties */
        union {
                __be32        imm_data;           /* Immediate data (in network byte order) *
                uint32_t      invalidate_rkey;    /* Remote rkey to invalidate */
        };
        union {
                struct {
                        uint64_t      remote_addr;   /* Start address of remote memo
                        uint32_t      rkey;          /* Key of the remote Memory Regio
                } rdma;
                struct {
                        uint64_t      remote_addr;   /* Start address of remote memo
                        uint64_t      compare_add;   /* Compare operand */
                        uint64_t      swap;          /* Swap operand */
                        uint32_t      rkey;          /* Key of the remote Memory Regio
                } atomic;
                struct {
                        struct ibv_ah  *ah;          /* Address handle (AH) for the rem
                        uint32_t      remote_qpn;    /* QP number of the destination
                        uint32_t      remote_qkey;   /* Q_Key number of the destina
                } ud;
        } wr;
        union {
                struct {
                        uint32_t remote_srqn;        /* Number of the remote SRQ *
                } xrc;
        } qp_type;
        union {
                struct {
                        struct ibv_mw          *mw;          /* Memory window (MW)
                        uint32_t               rkey;         /* The desired new rkey of the
                        struct ibv_mw_bind_info  bind_info;  /* MW additional bi
                } bind_mw;
                struct {
                        void                   *hdr;     /* Pointer address of inlin
```

```
                                                uint16_t        hdr_sz;  /* Inline header size */
                                                uint16_t        mss;     /* Maximum segment size for each
                                        } tso;
                        };
};

struct ibv_mw_bind_info {
                        struct ibv_mr         *mr;          /* The Memory region (MR) to bind the MW to */
                        uint64_t              addr;        /* The address the MW should start at */
                        uint64_t              length;       /* The length (in bytes) the MW should span */
                        int                   mw_access_flags; /* Access flags to the MW. Use ibv_access_flags */
};

struct ibv_sge {
                        uint64_t              addr;              /* Start address of the local memory buffer or number of bytes fro
                                                                  start of the MR for MRs which are IBV_ZERO_BASED */
                        uint32_t              length;             /* Length of the buffer */
                        uint32_t              lkey;               /* Key of the local Memory Region */
};
```

Each QP Transport Service Type supports a specific set of opcodes, as shown in the following table:

OPCODE	IBV_QPT_UD	IBV_QPT_UC	IBV_QPT_RC	IBV_QPT_XRC_SEND	IBV_QPT_RAW_PA
IBV_WR_SEND	X	X	X	X	X
IBV_WR_SEND_WITH_IMM	X	X	X	X	
IBV_WR_RDMA_WRITE		X	X	X	
IBV_WR_RDMA_WRITE_WITH_IMM		X	X	X	
IBV_WR_RDMA_READ			X	X	
IBV_WR_ATOMIC_CMP_AND_SWP			X	X	
IBV_WR_ATOMIC_FETCH_AND_ADD			X	X	
IBV_WR_LOCAL_INV		X	X	X	
IBV_WR_BIND_MW		X	X	X	
IBV_WR_SEND_WITH_INV		X	X	X	
IBV_WR_TSO	X				X

The attribute send_flags describes the properties of the WR. It is either 0 or the bitwise OR of one or more of the following flags:

IBV_SEND_FENCE Set the fence indicator. Valid only for QPs with Transport Service Type **IBV_QPT_RC**

IBV_SEND_SIGNALED Set the completion notification indicator. Relevant only if QP was created with sq_sig_all=0

IBV_SEND_SOLICITED Set the solicited event indicator. Valid only for Send and RDMA Write with immediate

IBV_SEND_INLINE Send data in given gather list as inline data
in a send WQE. Valid only for Send and RDMA Write. The L_Key will not be checked.

IBV_SEND_IP_CSUM Offload the IPv4 and TCP/UDP checksum calculation.
Valid only when **device_cap_flags** in device_attr indicates current QP is supported by checksum offload.

RETURN VALUE

ibv_post_send() returns 0 on success, or the value of errno on failure (which indicates the failure reason).

NOTES

The user should not alter or destroy AHs associated with WRs until request is fully executed and a work completion has been retrieved from the corresponding completion queue (CQ) to avoid unexpected behavior.

The buffers used by a WR can only be safely reused after WR the request is fully executed and a work completion has been retrieved from the corresponding completion queue (CQ). However, if the IBV_SEND_INLINE flag was set, the buffer can be reused immediately after the call returns.

IBV_WR_DRIVER1 is an opcode that should be used to issue a specific driver operation.

SEE ALSO

ibv_create_qp(3), **ibv_create_ah**(3), **ibv_post_recv**(3), **ibv_post_srq_recv**(3), **ibv_poll_cq**(3)

AUTHORS

Dotan Barak <dotanba@gmail.com>

Majd Dibbiny <majd@mellanox.com>

Yishai Hadas <yishaih@mellanox.com>