HERD KV存储简介

HERD uses a single round trip for all requests and supports up to 26 million key-value operations per second with 5 μs average latency.

- 每个Client和Server建立一个RDMA链接。

```
//common.h
void create_qp(struct ctrl_blk *ctx);
void modify_qp_to_init(struct ctrl_blk *ctx);
int setup_buffers(struct ctrl_blk *cb);
```

- Server上每一个Client都有一个对应的内存地址。Client端也注册了地址存放Server的回复。

```
//common.h
volatile struct KV *server_req_area;
volatile struct KV *server_resp_area;
volatile struct KV *client_req_area;
volatile struct UD_KV *client_resp_area;
```

- Client发送PUT Query或者GET Query时，使用RDMA WRITE将Query写入Server端对应的内存地址。

```
\\main.c
void run_client(struct ctrl_blk *cb){
    //...
    ret = ibv_post_send(cb->conn_qp[0], &cb->wr, &bad_send_wr);
    //...
}
```

- Server轮询本地对应每个Client的内存地址。

```
\\common.h
void poll_conn_cq(int num_completions, struct ctrl_blk *cb, int cq_num);
void poll_dgram_cq(int num_completions, struct ctrl_blk *cb, int cq_num);

\\main.c
void run_server(struct ctrl_blk *cb){
    //...
    int req_ind = req_lo[i] + (req_num[i] & WINDOW_SIZE_);
    if((char) server_req_area[req_ind].key[KEY_SIZE - 1] == 0) {
        failed_polls ++;
        if(failed_polls < FAIL_LIM) {
            continue;
        }
    }
    //...
}
```

- Server发现新的Query后，进行处理，然后用RDMA SEND将回复写入Client的内存中。

```
\\main.c
void run_server(struct ctrl_blk *cb){
    //...
    ret = ibv_post_send(cb->dgram_qp[0], &cb->wr, &bad_send_wr);
    //...
}
```