

Memcached Reference Design Specification

(Michaela Blott, Lisa Liu, Kimon Karras, Kees Vissers)

Rev 0.1

Revision History

Revision Number	Date	Author	Change Description
0.1	29/09/2014	Lisa Liu	Initial document
0.2	09/03/2015	Lisa Liu	Add memcached client specification

List of Reviewers

Name	Group	Reviewed Date

Contents

1. System Architecture.....	4
2. Memcached Client Usage	7
3. System Testing.....	8
3.1 FPGA Memcached Server Configuration	9
3.2 Memcached Client Configuration	9

List of Figures

Figure 1: Top-level Architecture	4
Figure 2: Memcached Pipeline Overview	5

1. System Architecture

The memcached application implemented on the alpha-data card includes a Key-Value-Store, a binary memcached protocol parser and formatter, a memory subsystem, a software memory management system on host PC and a UDP offload engine unit.

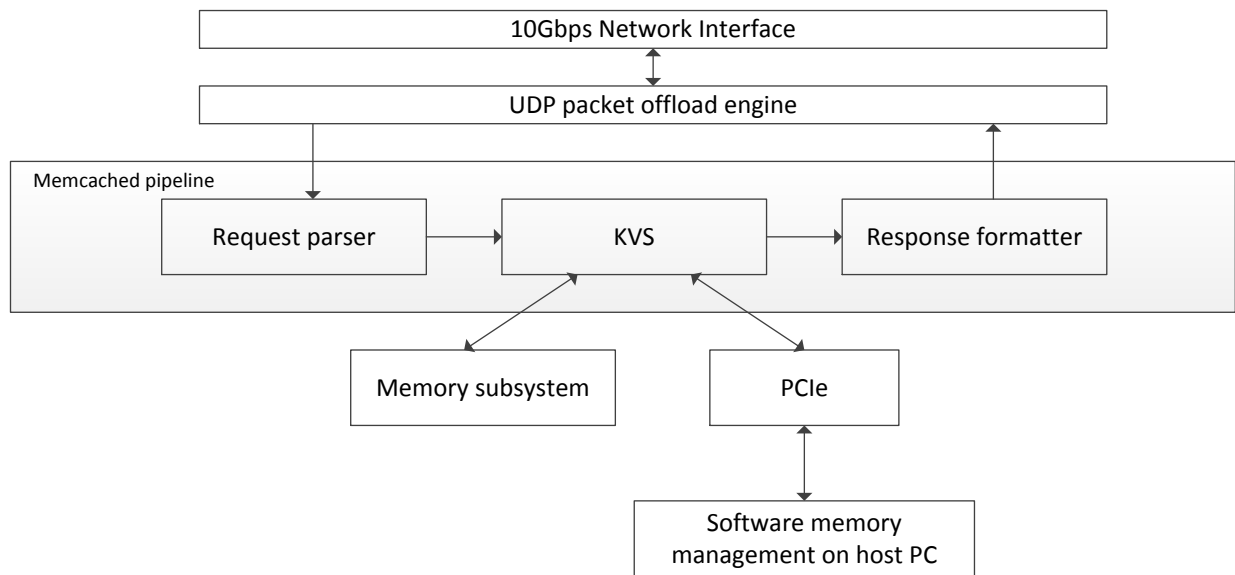


Figure 1: Top-level Architecture

The 10Gbps network interface includes layer 2 and below processing, as such it contains the MAC and potentially the PCS_PMA functionality. Packets received on the network interface are passed to the UDP offload engine, which extracts the payloads – memcached requests, and passes the requests to the memcached pipeline. The memcached pipeline analyzes the memcached packets, accesses the memory subsystem to store or retrieve key-value pairs and in the end formats the results into memcached packets to be sent back as UDP payloads to the network interface. On the host PC, a software memory management system is implemented to allocate and reclaim the memory addresses for the values.

The memcached pipeline is illustrated in more detail in figure 2. Basically, the request parser analyzes the memcached packets to extract keys, value, and meta-data information and generates an opcode for the currently supported subset of operations. Currently, only binary protocols are included in this reference design. The request parser normalizes all packets to the format of the standard axi stream interface. This information is then passed to the hash table. The hash table's responsibility is to produce an index into the value store for any incoming key. The value store simply supports read or write operations on a corresponding area of memory as defined by the opcode. For SET operations, the presented value is written into memory. In case of GET operations, the retrieved value is

added to the packet information as it streams to the response formatter. Finally, the response formatter supports formatting of responses according to the supported protocols.

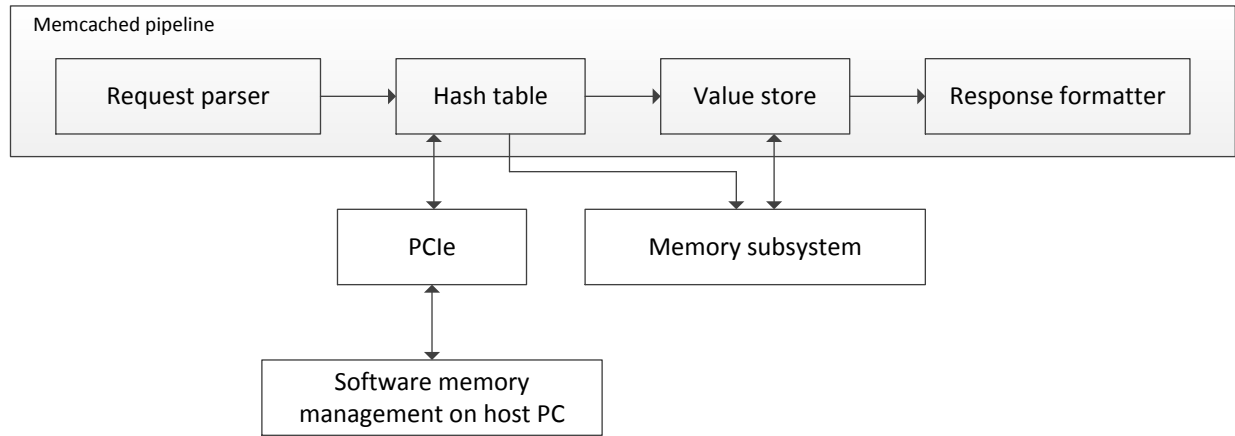
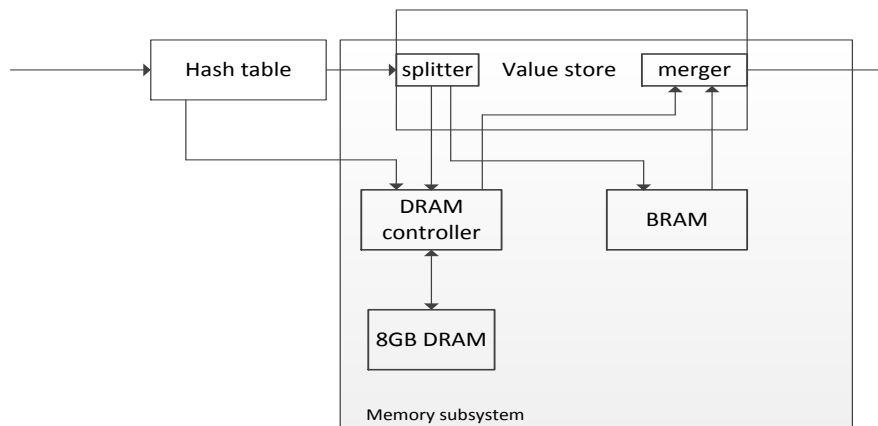


Figure 2: Memcached Pipeline Overview

To support multi-million searching entries, the current implementation uses DRAM to store the hash table and both DRAM and BRAM to store the values. The memory subsystem provides the standard axi stream interface to allow memcached pipeline to access different memory components. The memory addresses of values are allocated and deallocated by host software and accessed by hash table via PCIe link.

As illustrated in figure 3, the memory subsystem provides the storage for the hash table and value store. The value store is partitioned into two main areas, one that resides within DRAM and one within BRAM. A splitter and merger are needed whereby the splitter routes memory requests to either DRAM or BRAM on the basis of the size of the value and the merger simply aggregates the results back into the memcached pipeline data path.



© Copyright 2016 Xilinx

Figure 3: Memcached Subsystem Architecture

Figure 4 shows the architecture of the software memory management system, which is responsible for allocating and reclaiming the memory space for storing values. That is, when a new key-value pair arrives, this system allocates an address either in DRAM or in BRAM based on the value's size and inserts it into the hash table via PCIe link. Similarly, when items are removed, the freed address is returned to the memory management logic via PCIe link.

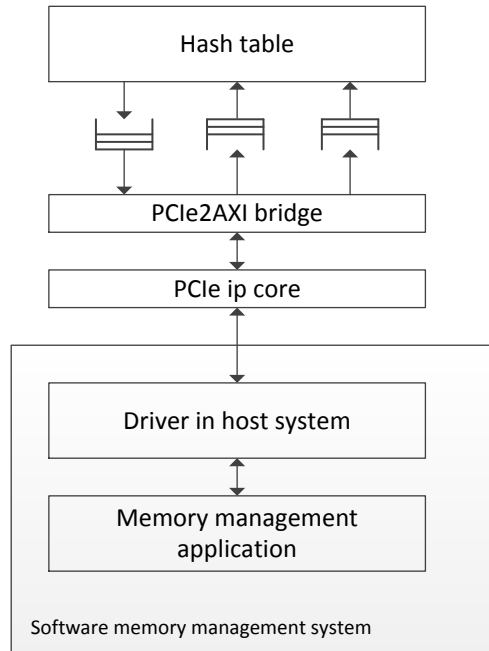


Figure 4: Memory Management System Architecture

2. Memcached Client Usage

To test the memcached FPGA server, a memcached client has been developed. The memcached client consists of two parts: requests generation and requests transmission. The requests generation part reads the key value size configuration file (.csv file) and generates corresponding memcached SET and GET requests. The requests transmission part reads the read_ratio and time_out options from command line, generates UDP packets that contains the memcached binary SET and GET requests as payload, and transmits the UDP packets according to the read_ratio until the time_out.

The usage of memcache client is given in following paragraphs.

USAGE: ./udp_mcd_client [OPTIONS]

--config_file_name	KV request configuration file name. Default=kv_pair.csv
--read_ratio	read ratio. Default=0.700000
--time_out	Default=1 seconds
--single_run	issue set and get requests generated from the config file once.
--help	print this message.

The key value configuration file (.csv) file lists all possible <key_size, value_size> pairs that are used in the client. **(Currently we only support value size <= 4096 bytes.)** For example, following text of a .csv file indicates there are 6 KV pairs used in the memcached client. The first pair contains one byte key and 8 byte value, the last pair contains 2 byte key and 896 byte value.

```
1,8
1,16
2,64
2,128
2,896
```

The “read_ratio” option defines the the percentage of the GET requests transmitted by the client. The “single_run” option causes the client to ignore “read_ratio” and “time_out” option, and issues SET request and GET requests for each KV pair specified in the configuration .csv file.

3. System Testing

To test the memcached FPGA server, please follow the steps below.

1. Deploy the UoeMcdSingleDramPCle_top.bit to the alpha data card via analyzer or implact or vivado hardware manager.
2. Reboot the Linux machine that contains the alpha data card. For the following reference, this Linux machine is called FPGA Memcached Server.
3. Copy the PCle_Driver.tar to the FPGA Memcached Server. Then, take following steps to start software memory management on the FPGA Memcached Server.
 - a. Navigate to the "PCle_Driver/Linux_Driver" directory
 - b. Run "sudo ./install_drive.sh to install PCle driver
 - c. Naviage to "PCle_Driver/memMgt" directory
 - d. Run " sudo make commandline_driver" to compile and build executable
 - e. Navigate to PCle_Driver/memMgt/bin/commandline_driver and run "sudo ./memMgt" to start memory management software
 - f. Once the memory management software starts, enter "d" to display the bitstream revision number. You should see "6".
 - g. Enter "m" to start memory management.
4. Connect the 10G interface of a Linux machine, which will be referred to as Memcached Client in the following paragraphs, to the FPGA Memcached Server.
5. On the Memcached Client, use ifconfig to set the ip address and netmask of the 10G interface that is connecting to alpha data card as:
1.1.1.x
255.255.255.0
6. Run "ifconfig 10G_eth_inf mtu 9000" to set the MTU of the 10G port on the Memcached client as 9000 bytes to avoid ip fragmentation.
7. Copy udp_mcd_client.tar to the Memcached Client, unzip the tar file and then navigate to udp_mcd_client and run "make" to build executable memcached client
8. On Memcached Client, run "ping 1.1.1.1" to see if the link between Memcached Client and Server is up.
9. On Memcached Client, start wireshark to capture the packets for the 10G prot, run ". /udp_mcd_client --config_file_name kv_config.csv --single_run", then you should be able to see something like follow image.

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help						
Filter: Expression... Clear Apply Save TCP Ports						
No.	Time	Source	Destination	Protocol	Length	Info
30	57.782376000	1.1.1.1	1.1.1.20	MEMCACHE	66	Set Response
31	57.782388000	1.1.1.20	1.1.1.1	MEMCACHE	3151	Set Request
32	57.782404000	1.1.1.1	1.1.1.20	MEMCACHE	66	Set Response
33	57.782416000	1.1.1.20	1.1.1.1	MEMCACHE	4176	Set Request
34	57.782434000	1.1.1.1	1.1.1.20	MEMCACHE	66	Set Response
35	57.782448000	1.1.1.20	1.1.1.1	MEMCACHE	67	Get Request
36	57.782458000	1.1.1.1	1.1.1.20	MEMCACHE	78	Get Response
37	57.782471000	1.1.1.20	1.1.1.1	MEMCACHE	67	Get Request
38	57.782480000	1.1.1.1	1.1.1.20	MEMCACHE	86	Get Response
39	57.782492000	1.1.1.20	1.1.1.1	MEMCACHE	67	Get Request
40	57.782501000	1.1.1.1	1.1.1.20	MEMCACHE	102	Get Response
41	57.782512000	1.1.1.20	1.1.1.1	MEMCACHE	68	Get Request
42	57.782524000	1.1.1.1	1.1.1.20	MEMCACHE	134	Get Response
43	57.782538000	1.1.1.20	1.1.1.1	MEMCACHE	68	Get Request
44	57.782548000	1.1.1.1	1.1.1.20	MEMCACHE	198	Get Response
45	57.782562000	1.1.1.20	1.1.1.1	MEMCACHE	68	Get Request
46	57.782577000	1.1.1.1	1.1.1.20	MEMCACHE	326	Get Response
47	57.782590000	1.1.1.20	1.1.1.1	MEMCACHE	68	Get Request
48	57.782602000	1.1.1.1	1.1.1.20	MEMCACHE	454	Get Response
49	57.782613000	1.1.1.20	1.1.1.1	MEMCACHE	68	Get Request
50	57.782626000	1.1.1.1	1.1.1.20	MEMCACHE	582	Get Response
51	57.782637000	1.1.1.20	1.1.1.1	MEMCACHE	68	Get Request
52	57.782650000	1.1.1.1	1.1.1.20	MEMCACHE	838	Get Response
53	57.782661000	1.1.1.20	1.1.1.1	MEMCACHE	68	Get Request
54	57.782674000	1.1.1.1	1.1.1.20	MEMCACHE	966	Get Response
55	57.782685000	1.1.1.20	1.1.1.1	MEMCACHE	69	Get Request
56	57.782698000	1.1.1.1	1.1.1.20	MEMCACHE	1094	Get Response
57	57.782710000	1.1.1.20	1.1.1.1	MEMCACHE	70	Get Request
58	57.782728000	1.1.1.1	1.1.1.20	MEMCACHE	2118	Get Response
59	57.782740000	1.1.1.20	1.1.1.1	MEMCACHE	71	Get Request
60	57.782761000	1.1.1.1	1.1.1.20	MEMCACHE	3142	Get Response
61	57.782774000	1.1.1.20	1.1.1.1	MEMCACHE	72	Get Request

- On Memcached Client, stop wireshark, and run “./udp_mcd_client –config_file_name kv_config.csv –read_ratio 0.9 –time_out 60” to run memcached client for 60 seconds with given read ratio.

3.1 FPGA Memcached Server Configuration

Operating System:

Linux xird157.xilinx.com 3.14.22-100.fc19.x86_64 #1 SMP Wed Oct 15 12:46:05 UTC 2014
x86_64 x86_64 x86_64 GNU/Linux

3.2 Memcached Client Configuration

Operating System:

Linux xirlabsws4 3.17.8-200.fc20.x86_64 #1 SMP Thu Jan 8 23:26:57 UTC 2015 x86_64 x86_64
x86_64 GNU/Linux