Dataflow Architectures for 10Gbps Line-rate Key-value-Stores

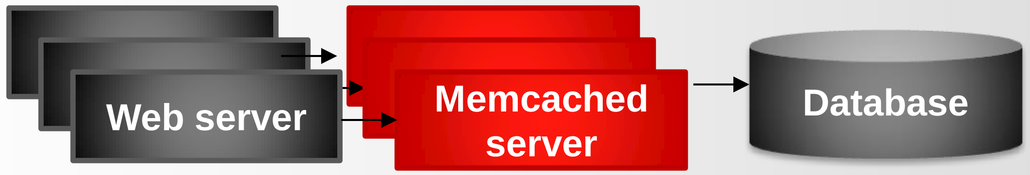
Michaela Blott, Kees Vissers - Xilinx Research

Agenda

- Current key-value stores (KVS)
  - State-of-the-art
  - Bottlenecks
- Dataflow architectures for KVS
  - Why dataflow architectures
  - Prototype architecture
  - Results
  - Limitations

Key-Value Stores

- Common middleware application to alleviate access bottlenecks on databases



- Most popular and most recent database contents are cached in main memory of a tier of x86 servers

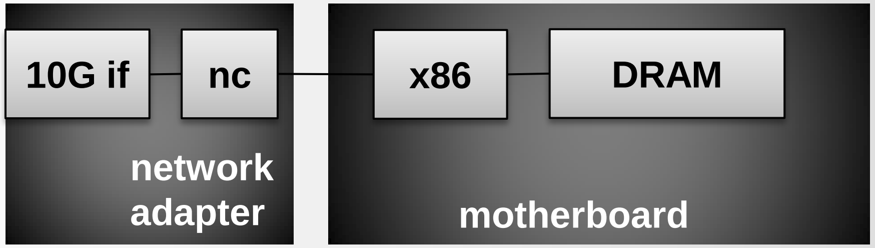
- Provides the abstraction of an associative memory
  - Values are stored or retrieved by sending the associated key
  - GET(KEY) and SET(KET,VALUE)

```
GET(k):
    receive(p);
    k = parse(p);
    a = hashKey(k);
    v = readValue(a);
    new_p = format(v);
    send(new_p);
```

- Memcached is a commonly used open source package for KVS

Typical Implementations

- **Hardware:**



- **Software**
  - Each connection is represented as a struct (c)
  - Any event on the connection state is distributed to pthreads (via Libevent)
  - All worker threads run the same code (drive_machine())
    - Loop over switch statement over the connection state
    - Locks on sockets, hash table, and value store areas/items

Thread 0 … thread n-1

```
drive_machine():
while (!stop) {
   switch(c->state) {
      case connection_waiting:
      case connection_closing:
       …
      case new_command:
         lock socket;
         read from socket;
         unlock socket;
         parse;
      case read_htable:
         hash key;
         lock hash table;
         hash table access;
         hash table LRU;
         unlock hash table;
      case write_output:
      …
```

Best Published Performance Numbers

Platform	RPS [M]	Latency [us]	RPS/W [K]
Intel® Xeon® (8 cores)*	1.34	200-300	7
Intel Xeon (2 sockets, 16 cores)*	3.15	200-300	11.2
Memcached with Infiniband & Intel Xeon (2 sockets, 16cores)**	1.8	12	Unknown
TilePRO (64 cores)***	0.34	200-400	3.6
TilePRO (4x64 cores)***	1.34	200-400	5.8
Chalamalasetti (FPGA)****	0.27	2.4-12	30.04

7K

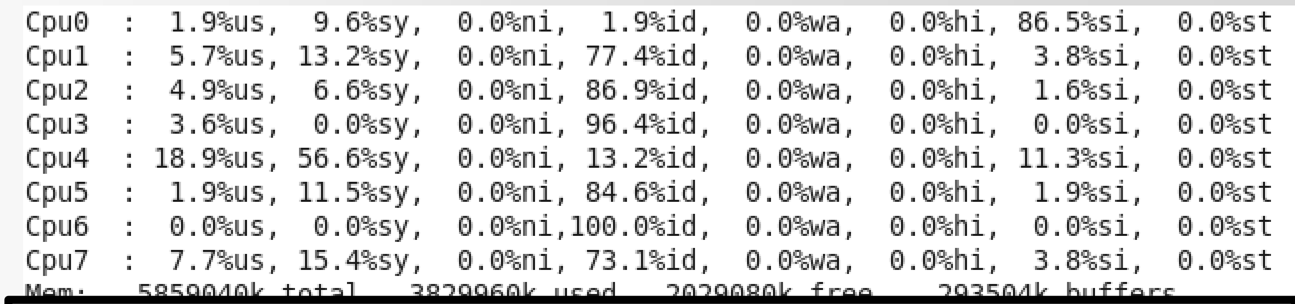
1.4MRPS

200us latency

* WIGGINS, A., AND LANGSTON, J. Enhancing the scalability of memcached. In Intel Software Network (2012).
**JOSE, J., SUBRAMONI, H., LUO, M., ZHANG, M., HUANG, J., UR RAHMAN, M. W., ISLAM, N. S., OUYANG, X., WANG, H., SUR, S., AND PANDA, D. K. Memcached design on high performance rdma capable interconnects. 2012 41st International Conference on Parallel Processing 0 (2011), 743–752.
*** BEREZECKI, M., FRACHTENBERG, E., PALECZNY, M., AND STEELE, K. Power and performance evaluation of memcached on the tilepro64 architecture. In Green Computing Conference and Workshops (IGCC), 2011 International (July 2011), pp. 1 –8.
**** Kevin Lim, David Meisner, Ali G. Saidi, Parthasarathy Ranganathan, and Thomas F. Wenisch. 2013. Thin servers with smart pipes: designing SoC accelerators for memcached. In*Proceedings of the 40th Annual International Symposium on Computer Architecture* (ISCA '13). ACM, New York, NY, USA, 36-47.

Bottlenecks – TCP/IP Stack

- **CPU intensive**



```
Cpu0  :  1.9%us,  9.6%sy,  0.0%ni,  1.9%id,  0.0%wa,  0.0%hi, 86.5%si,  0.0%st
Cpu1  :  5.7%us, 13.2%sy,  0.0%ni, 77.4%id,  0.0%wa,  0.0%hi,  3.8%si,  0.0%st
Cpu2  :  4.9%us,  6.6%sy,  0.0%ni, 86.9%id,  0.0%wa,  0.0%hi,  1.6%si,  0.0%st
Cpu3  :  3.6%us,  0.0%sy,  0.0%ni, 96.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 18.9%us, 56.6%sy,  0.0%ni, 13.2%id,  0.0%wa,  0.0%hi, 11.3%si,  0.0%st
Cpu5  :  1.9%us, 11.5%sy,  0.0%ni, 84.6%id,  0.0%wa,  0.0%hi,  1.9%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  7.7%us, 15.4%sy,  0.0%ni, 73.1%id,  0.0%wa,  0.0%hi,  3.8%si,  0.0%st
Mem:   5859040k total,  3829960k used,  2029080k free,   293504k buffers
```

```
Total: 45%us, 113%sy, 0%ni,  534%id,  0%wa,  0%hi, 109%si,  0%st
```

*Top (while running 4 memcached instances

- **Frequent interrupts***
  - Leads to high rate of instruction cache misses (up to 160 MPKI)
    - Requires much larger L1 instruction caches
  - Causes poor branch predictability
    - Stalls in the superscalar pipeline architecture of standard x86

CPI: 2.5

- **High latency***
  - Packets have to be DMA'ed from/to network adapter over the PCIe® bus which introduces high latency

=> No resource sharing between memcached and TCP/IP stack
=> Close integration of network, compute and memory

XILINX ALL PROGRAMMABLE.

Bottlenecks –
Synchronization Overhead and L3 Cache

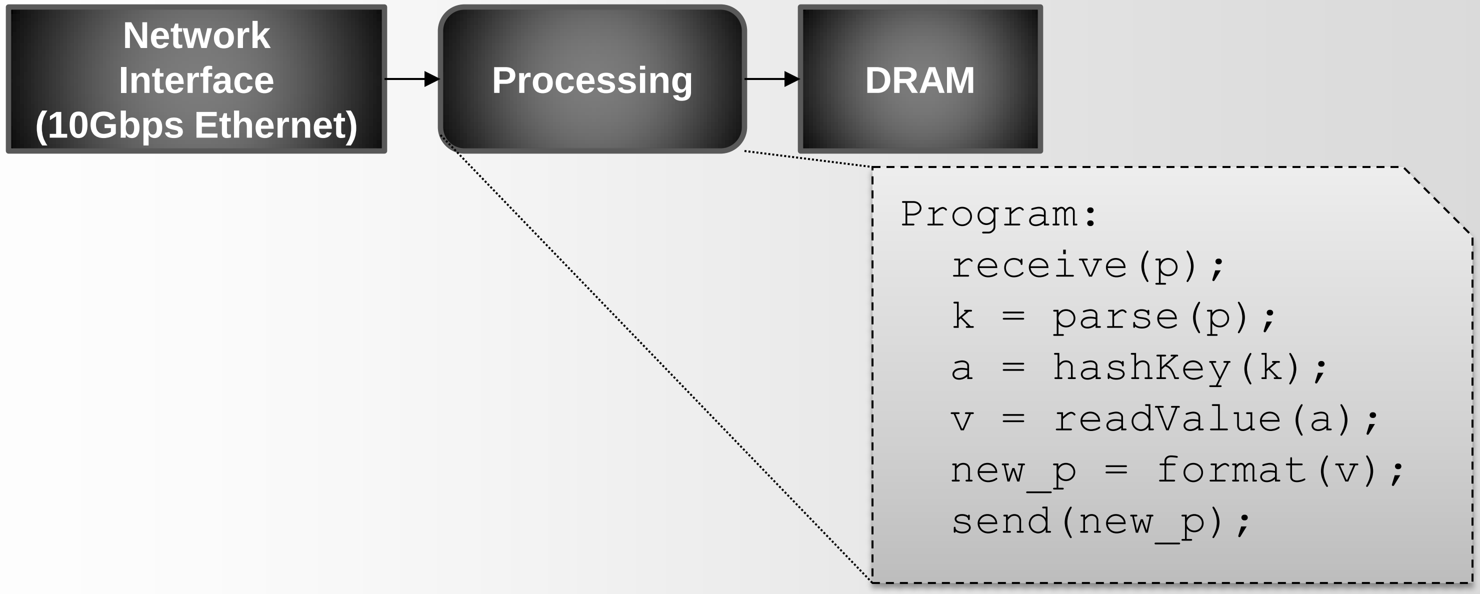
- **Threads stall on memory locks**
  1. Large locks effectively serialize execution
  2. Synchronization races cause poor branch predictability
  - This leads to inefficient use of superscalar pipelines
  - Intel has shown that by improving the granularity of the locks, we can scale to 1.4 MRPS (from 0.2MRPS)*

- **Last level cache ineffective due to random-access nature of key-value-stores (miss rate 60% - 95%**)**
  - Multithreading can't effectively hide memory access latencies
  - Cause considerable power waste

=> Exploitation of instruction-level parallelism through data-flow architectures
=> Static memory access schedule eliminates memory arbitration conflict
=> Data caching is ineffective

Why Dataflow Architectures?

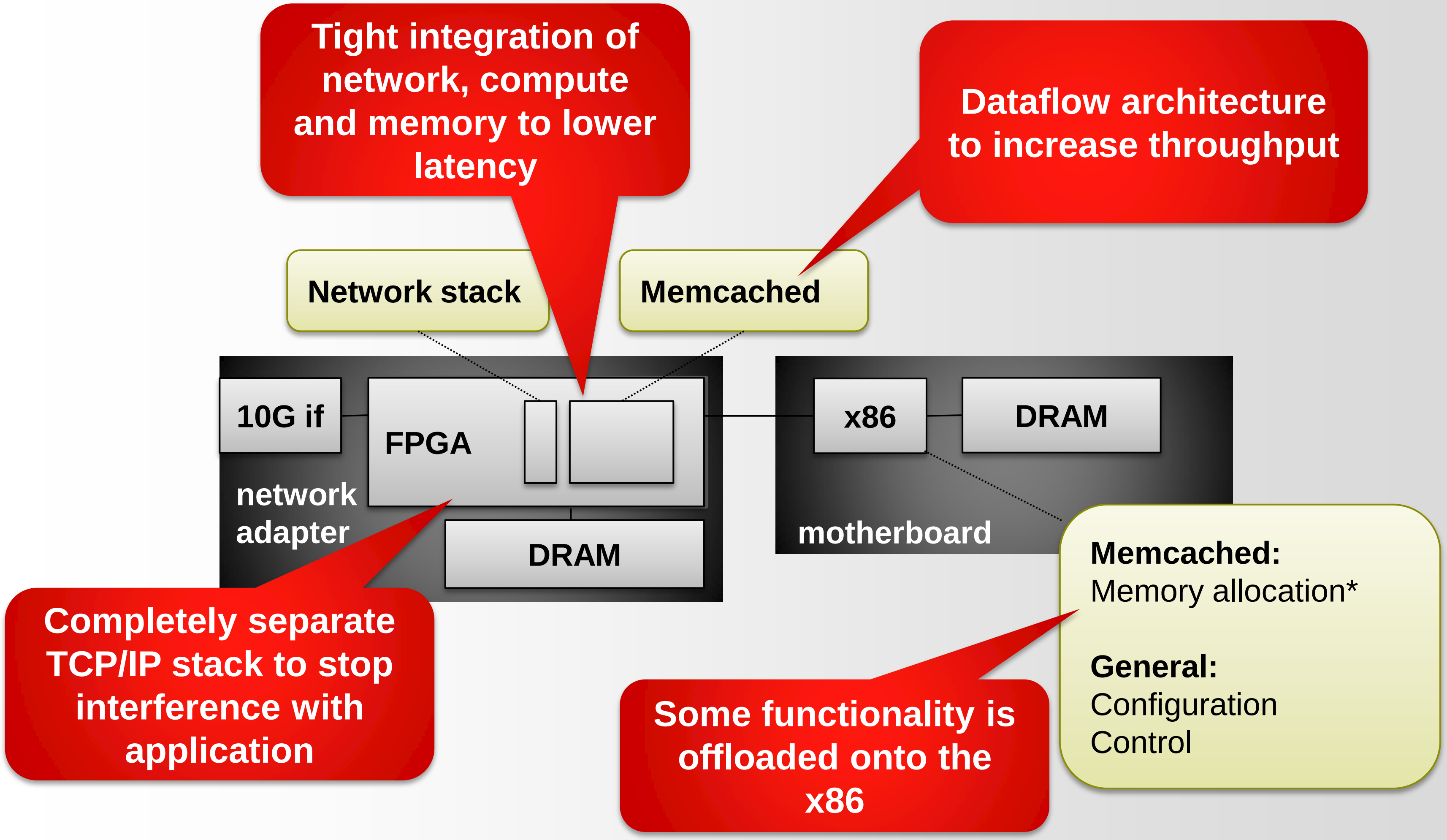
- **Memcached is fundamentally a streaming problem**
  - Data is moved from network to memory and back with little compute



```
Program:
  receive(p);
  k = parse(p);
  a = hashKey(k);
  v = readValue(a);
  new_p = format(v);
  send(new_p);
```

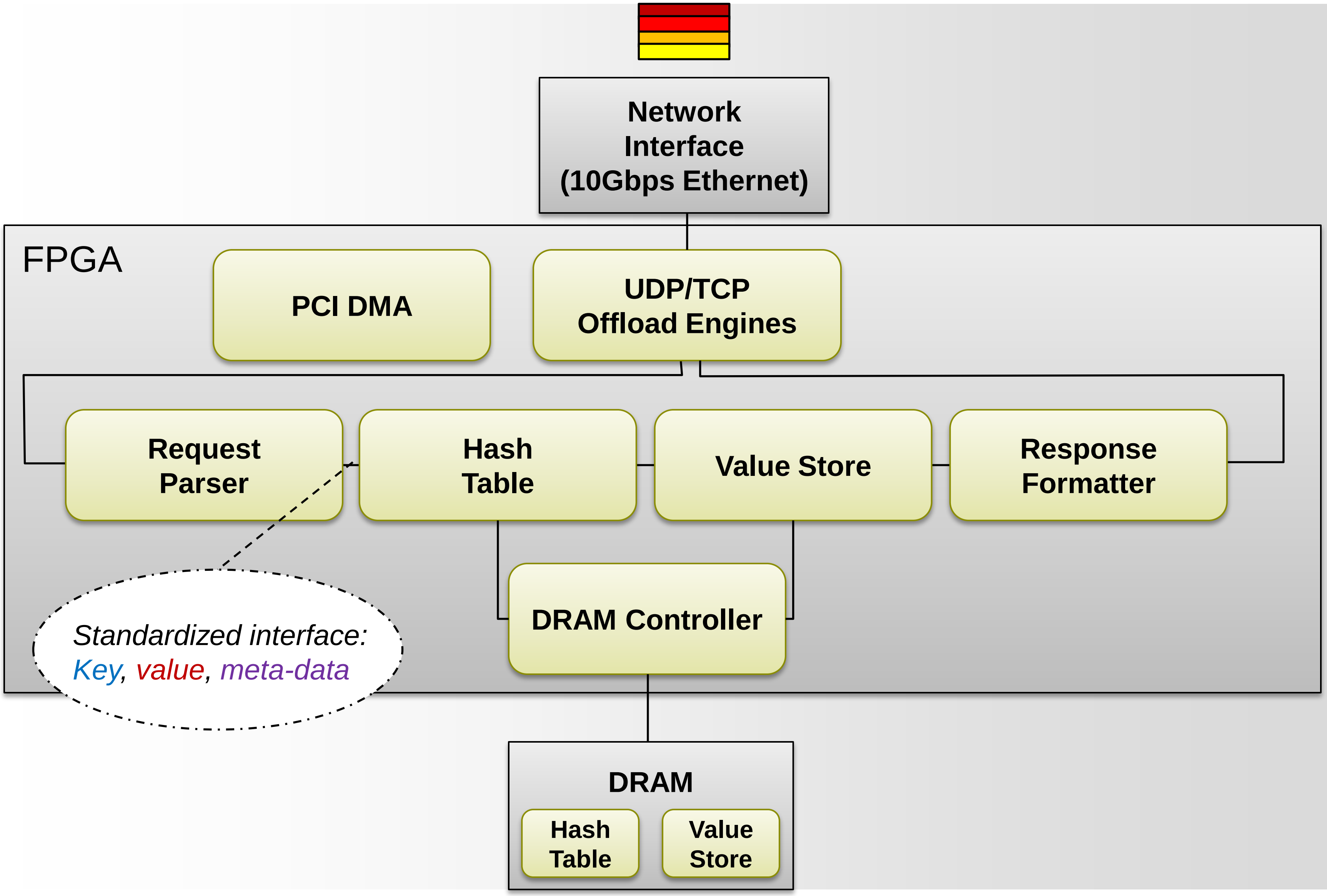
- **As such, dataflow architectures, frequently used for network processing, should be well suited towards the application**
  - Higher performance
  - Lower power consumption

Proposed System Architecture

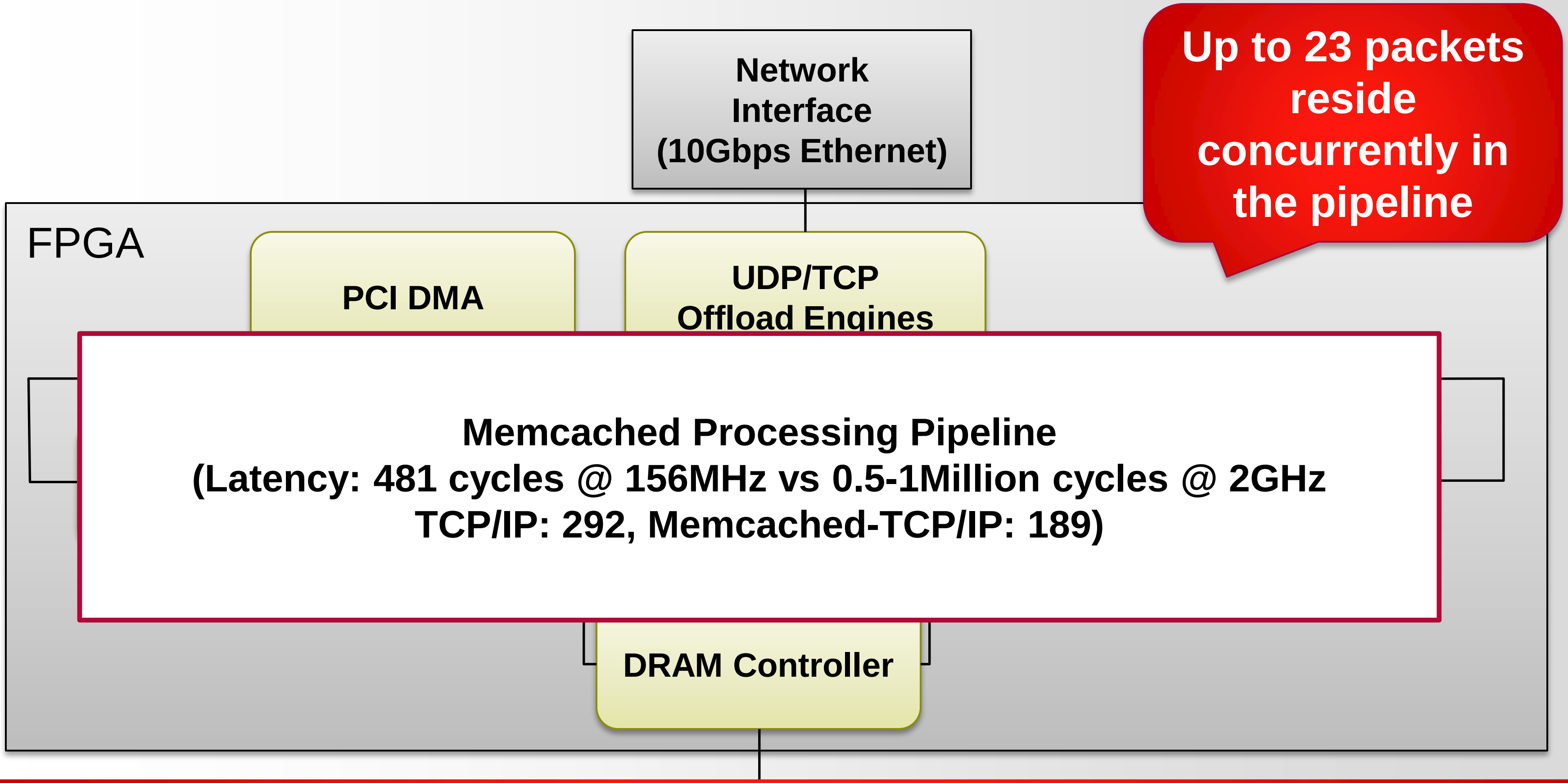

Tight integration of network, compute and memory to lower latency
Dataflow architecture to increase throughput
Network stack
Memcached
10G if
FPGA
network adapter
DRAM
x86
DRAM
motherboard
Memcached:
Memory allocation*
General:
Configuration
Control
Completely separate TCP/IP stack to stop interference with application
Some functionality is offloaded onto the x86

*below 3% of 1 core for 10% SET operations
*limited memory access bandwidth on platform

FPGA-based Dataflow Architecture



FPGA-based Dataflow Architecture

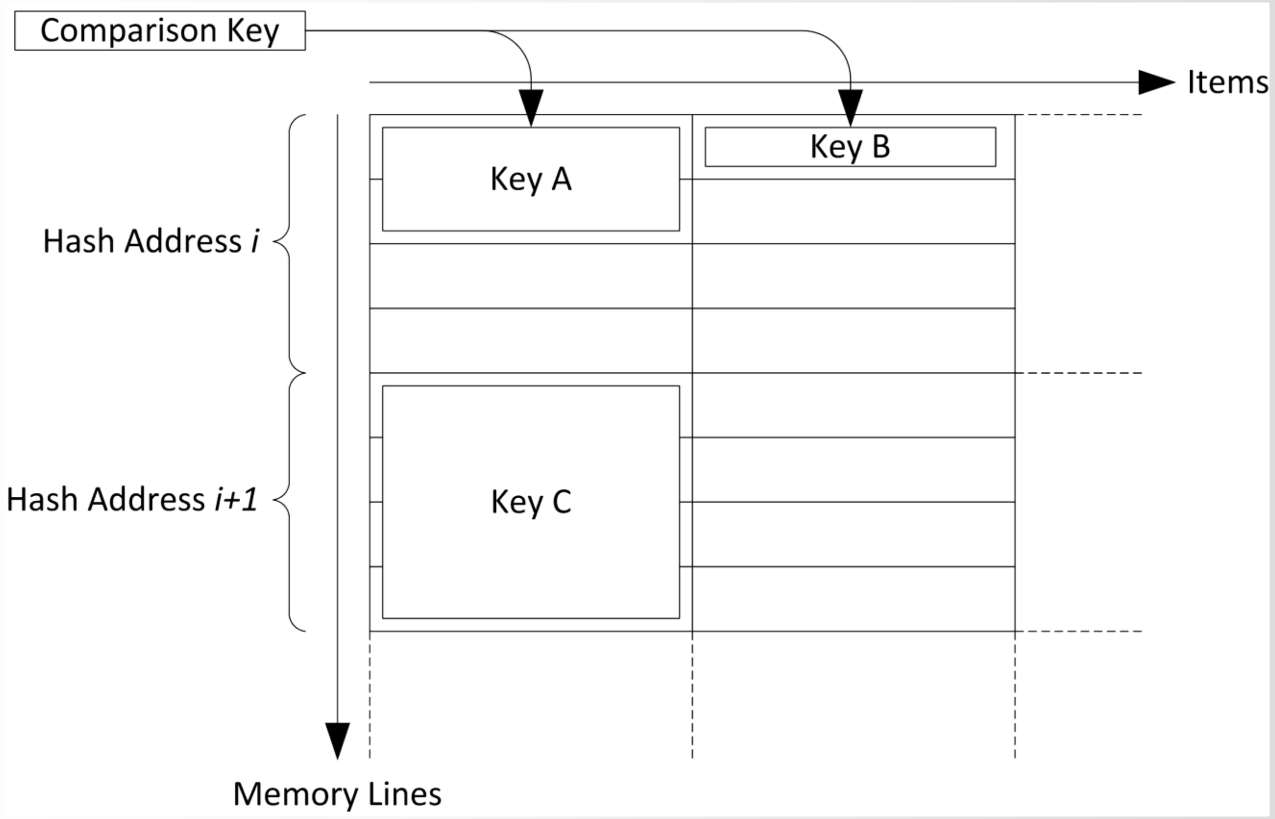

Network Interface (10Gbps Ethernet)
FPGA
PCI DMA
UDP/TCP Offload Engines
DRAM Controller
Up to 23 packets reside concurrently in the pipeline

Memcached Processing Pipeline
(Latency: 481 cycles @ 156MHz vs 0.5-1Million cycles @ 2GHz
TCP/IP: 292, Memcached-TCP/IP: 189)

=> Exploiting instruction-level parallelism increases throughput, lowers latency and is more power efficient
=> Inherently scalable

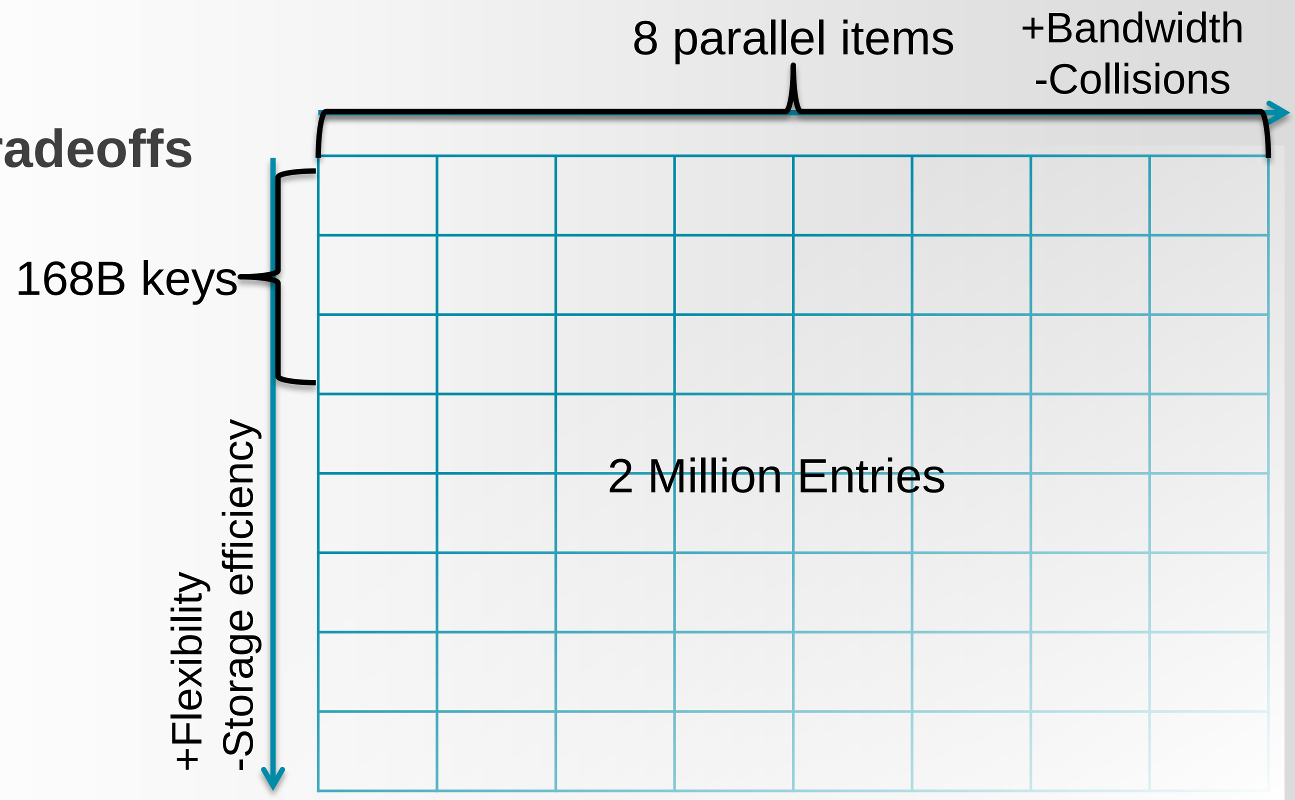
Hash Table Architecture

- **Collision handling through parallel lookup (8-way)**
  - Suits the wide memory bus
- **Flexible key handling through striping**



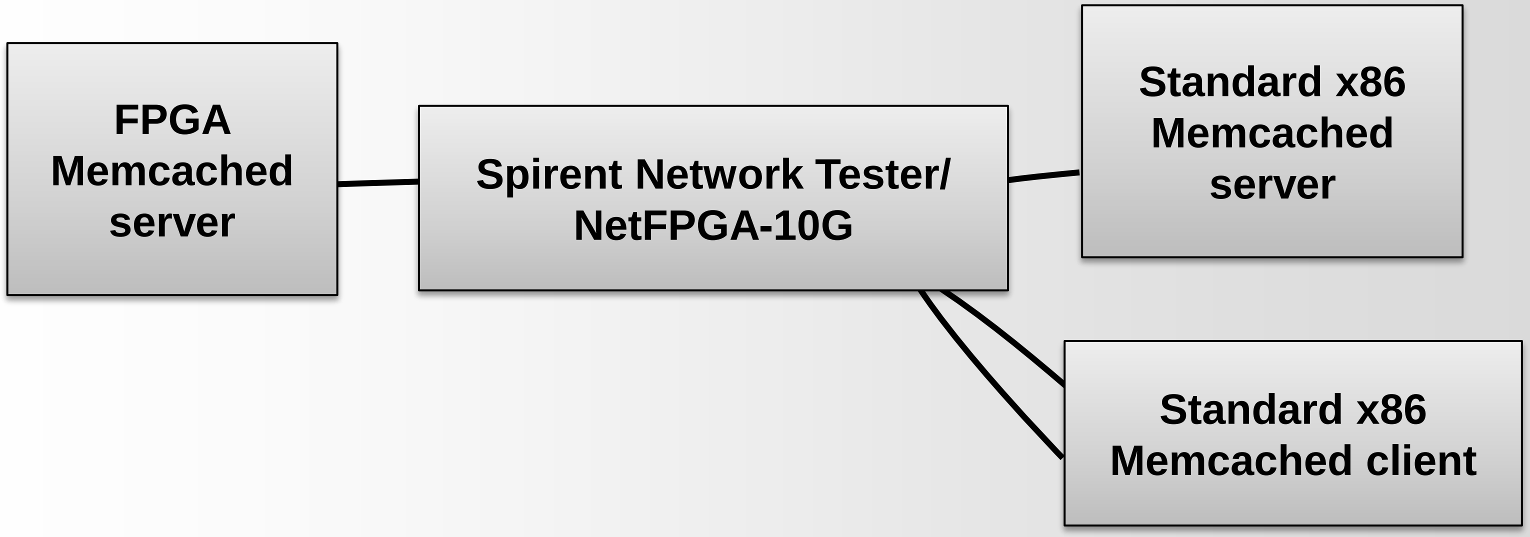
Hash Table Dimensions

- **Design tradeoffs**

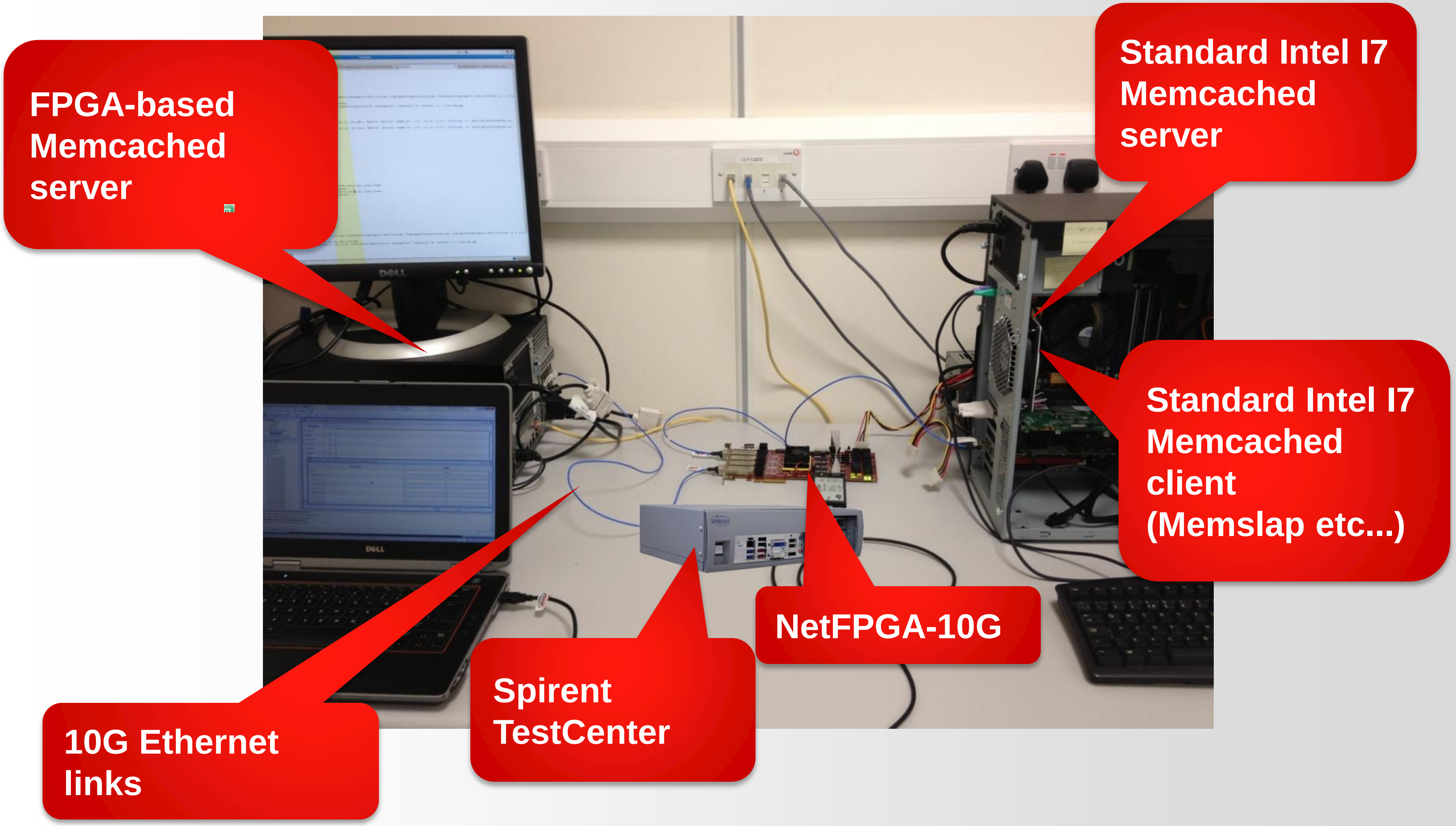


- **Size for hash table <400MB**
  - Key limit is 168 byte (memcached max: 250B, most use-cases <50B)
- **On our platform this hash table manages 23.6GB of value storage**

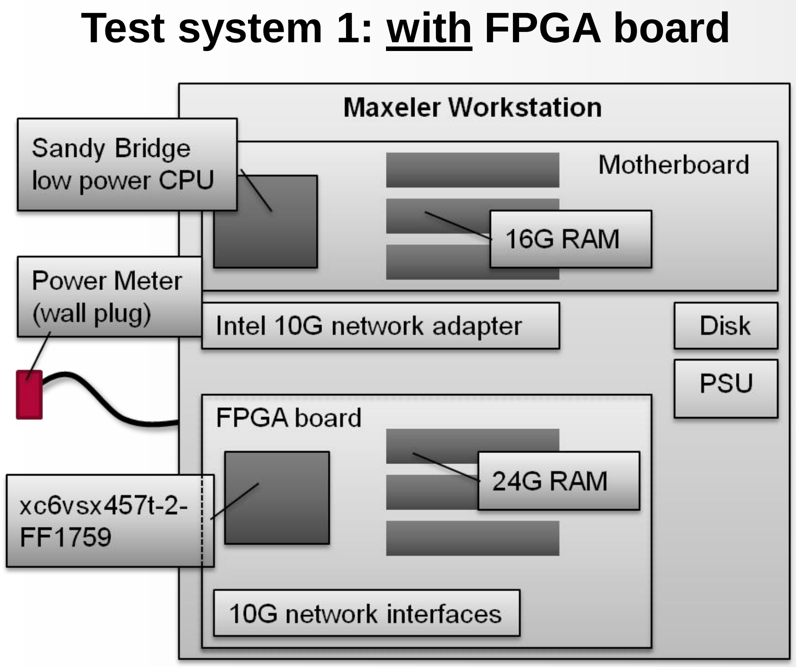
System Test Setup

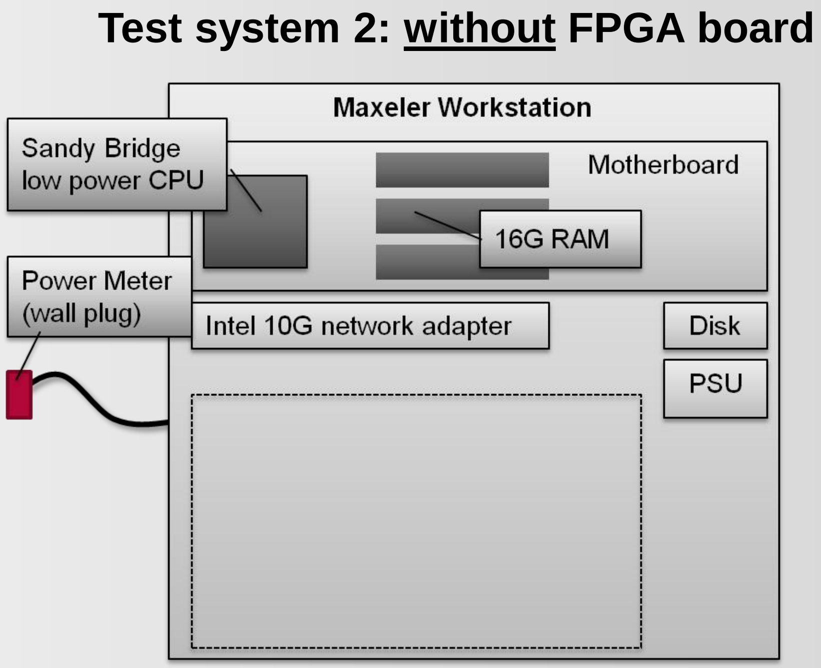

FPGA Memcached server
Spirent Network Tester/ NetFPGA-10G
Standard x86 Memcached server
Standard x86 Memcached client

System Test Setup


FPGA-based Memcached server
Standard Intel I7 Memcached server
Standard Intel I7 Memcached client (Memslap etc...)
NetFPGA-10G
Spirent TestCenter
10G Ethernet links

Power - Test Setup & Results

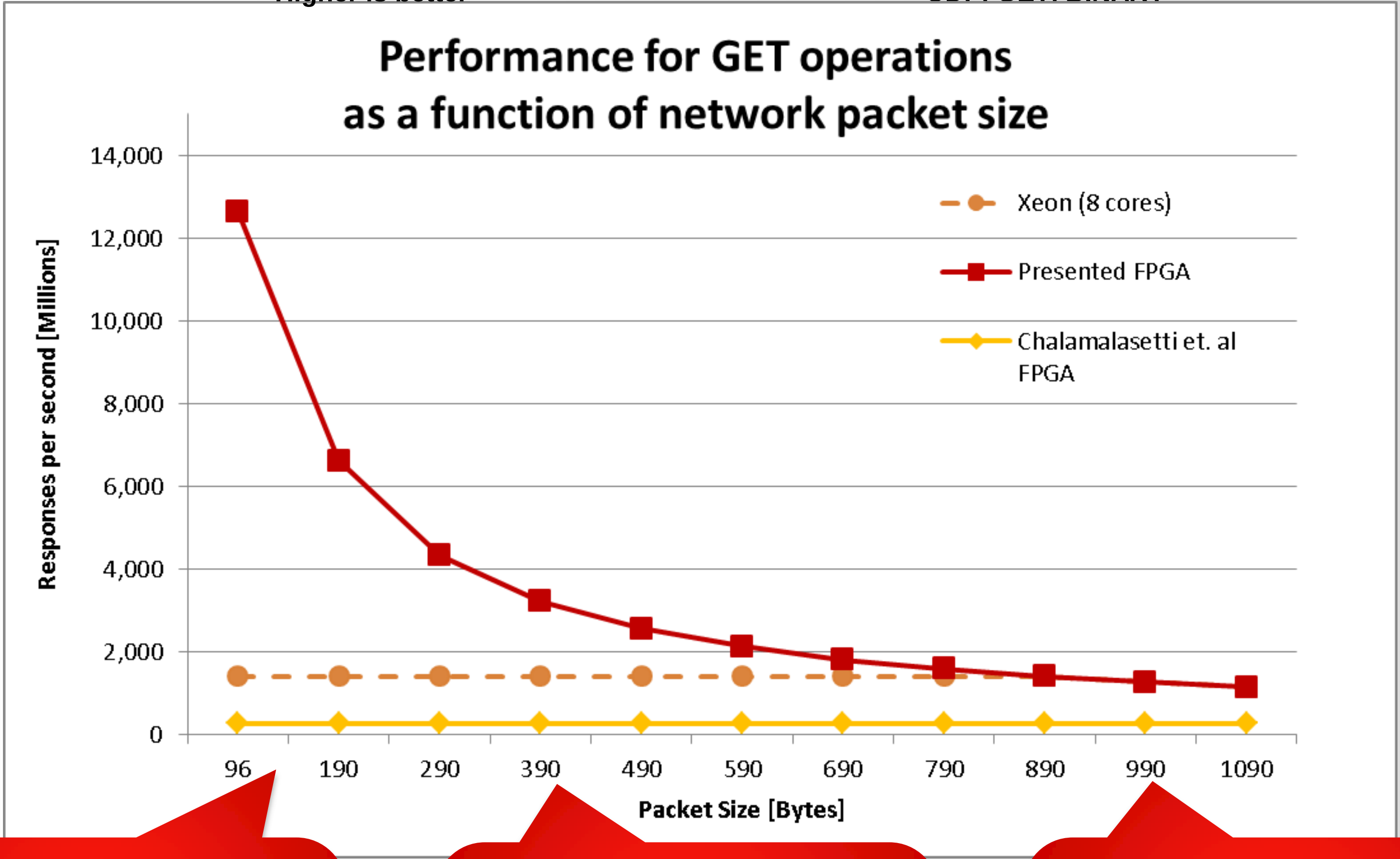



Test system 2: without FPGA board
Maxeler Workstation
Sandy Bridge low power CPU
Motherboard
16G RAM
Power Meter (wall plug)
Intel 10G network adapter
Disk
PSU

**(Power sourced from: power plug meter, xpower, data sheets and power regulator readings)*
***(UDP, binary protocol)*
****(includes FPGA and host system)*

Results - Performance

Higher is better *** UDP, GET, BINARY**



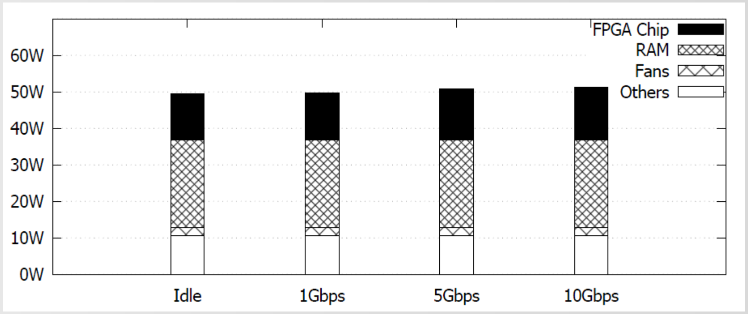
FPGA delivers constant 10Gbps performance – network becomes the bottleneck

X86 performance limited by a per packet overhead

Set performance saturates network as well

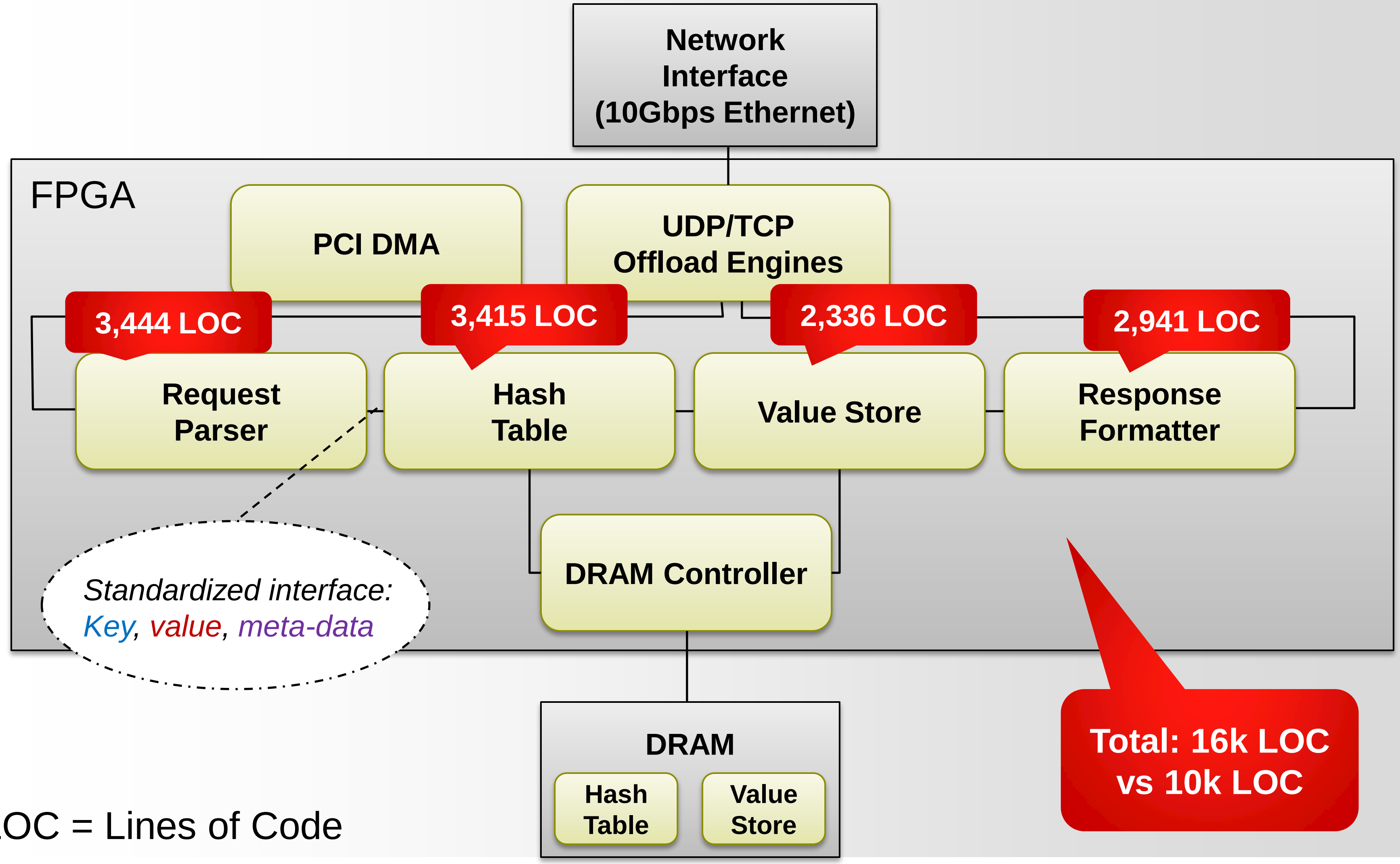
First Results of Memcached Evaluation

- **Sustained line rate processing for 10GE – 13MRPS possible, at smallest packet size**
  - Significant improvement over latest x86 numbers
- **Lower power**
- **Combined: 36x in RPS/Watt with low variation**
- **Cutting edge latency**
  - microseconds instead of 100s of microseconds



Platform	**RPS [M]**	**Latency [us]**	**RPS/W [K]**
Intel Xeon (8 cores)	1.34	200-300	7
TilePRO (64 cores)	0.34	200-400	3.6
FPGA (board only)	13.02	3.5-4.5	254.8
FPGA (with host)	13.02	3.5-4.5	106.7

Code Complexity

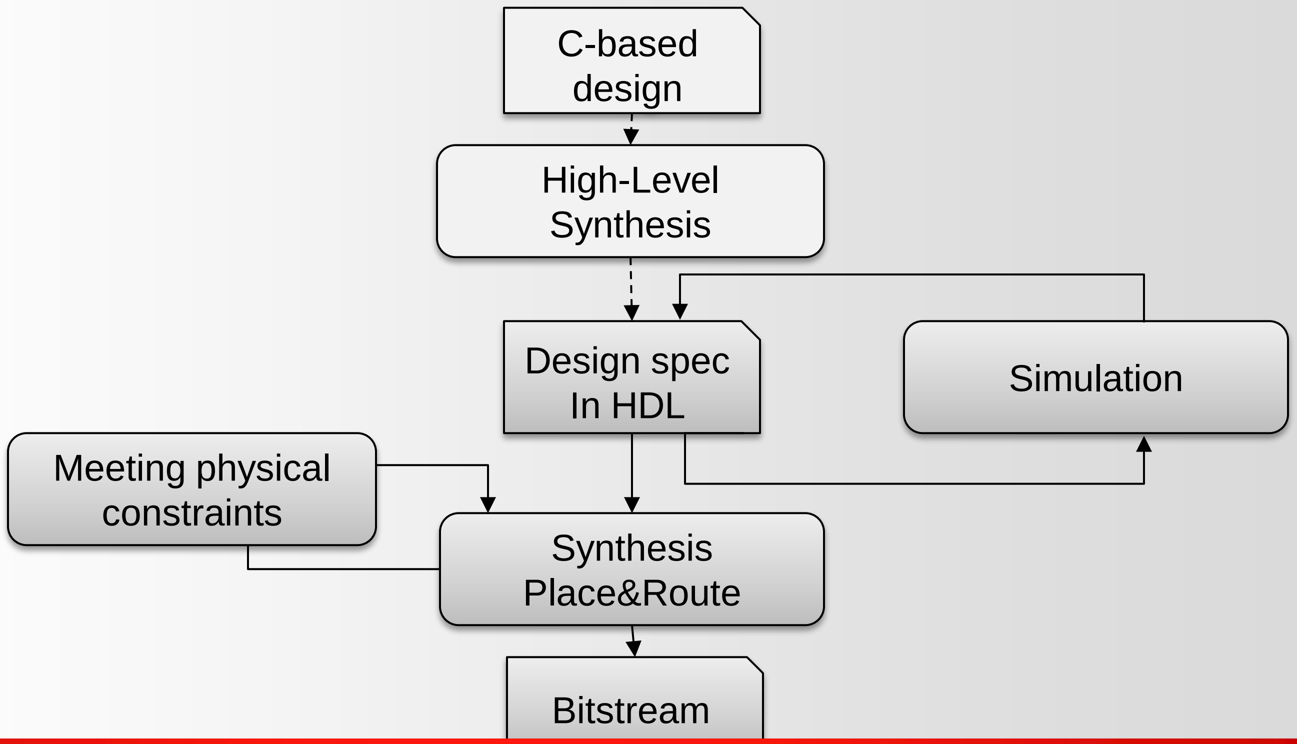


LOC = Lines of Code

Limitations

Development Effort

- Hardware design exposes a greater complexity to the user and requires therefore more engineering effort
- HLS reduces the complexity and shortens the development effort

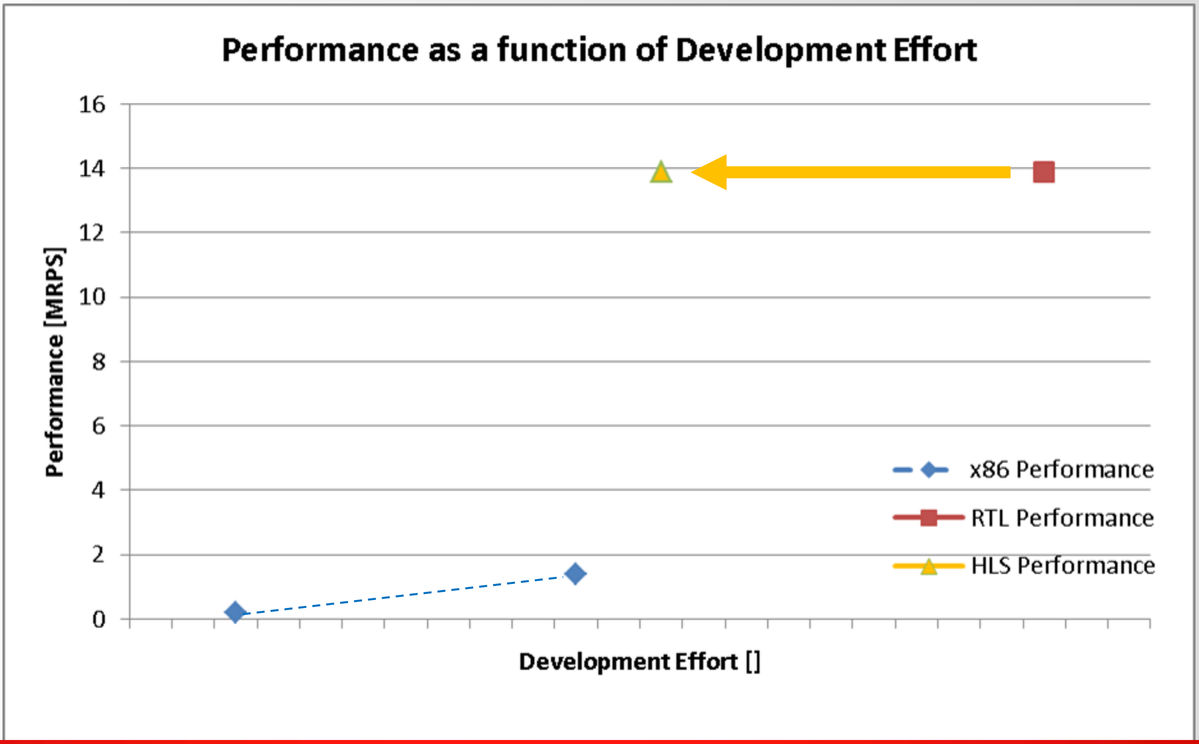


=> Greater performance at expense of larger development effort
=> Exploration of how HLS can reduce the cost

Limitations

Development Effort

- Hardware design exposes a greater complexity to the user and requires therefore more engineering effort
- HLS reduces the complexity and shortens the development effort



=> Greater performance at expense of larger development effort
=> Exploration of how HLS can reduce the cost

Other Limitations

- TCP offload restricted to #sessions

  => Future investigation into high session count TOE

- Limited storage capacity

  => SSD

- Memory allocation & cache management on host CPU

  Limited collision handling

  Limited protocol support

  => Exploration of SoC architecture

Summary & Next Steps

- Dataflow architecture delivers 10Gbps line-rate performance and scalability to higher rates
- Significantly higher RPS/Watt, with that lower TCO
- Minimal latency
- HLS reduces the complexity and shortens the development effort
- Next Steps:
  - Address limitations
  - Trials with real use cases

XILINX
ALL PROGRAMMABLE™

Thank You.
mblott@xilinx.com

References

[1] ARVIND, AND NIKHIL, R. Executing a program on the mittagged-token dataflow architecture. Computers, IEEE Transactions on 39, 3 (March 1990), 300–318.

[2] ATIKOGLU, B., XU, Y., FRACHTENBERG, E., JIANG, S., AND PALECZNY, M. Workload analysis of a large-scale key-value store. SIGMETRICS Perform. Eval. Rev. 40, 1 (June 2012), 53–64.

[3] BANDO, M., ARTAN, N., AND CHAO, H. Flashlook: 100-gbps hash-tuned route lookup architecture. In High Performance Switching and Routing, 2009. HPSR 2009. International Conference on (June 2009), pp. 1 –8.

[4] BEREZECKI, M., FRACHTENBERG, E., PALECZNY, M., AND STEELE, K. Power and performance evaluation of memcached

on the tilepro64 architecture. In Green Computing Conference and Workshops (IGCC), 2011 International (July 2011), pp. 1 –8.

[5] CONG, J., LIU, B., NEUENDORFFER, S., NOGUERA, J., VISSERS, K., AND ZHANG, Z. High-level synthesis for fpgas:

From prototyping to deployment. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 30, 4 (April 2011), 473 –491.

[6] FERDMAN, M., ADILEH, A., KOCBERBER, O., VOLOS, S., ALISAFAEE, M., JEVDJIC, D., KAYNAK, C., POPESCU, A. D., AILAMAKI, A., AND FALSAFI, B. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. SIGARCH Comput. Archit. News 40, 1 (Mar. 2012), 37–48.

[7] HETHERINGTON, T. H., ROGERS, T. G., HSU, L., O'CONNOR, M., AND AAMODT, T. M. Characterizing and evaluating a key-value store application on heterogeneous cpu-gpu systems. In Proceedings of the 2012 IEEE International Symposium on Performance Analysis of Systems & Software (Washington, DC,

USA, 2012), ISPASS '12, IEEE Computer Society, pp. 88–98.

[8] ISTVAN, Z. Hash Table for Large Key-Value Stores on FPGAs. Master's thesis, ETH Zurich, Dept. of Computer Science, Systems Group, Switzerland, 2013.

[9] JOSE, J., SUBRAMONI, H., LUO, M., ZHANG, M., HUANG, J., UR RAHMAN, M. W., ISLAM, N. S., OUYANG, X., WANG, H., SUR, S., AND PANDA, D. K. Memcached design on high performance rdma capable interconnects. 2012 41st International Conference on Parallel Processing 0 (2011), 743–752.

[10] LANG, W., PATEL, J. M., AND SHANKAR, S. Wimpy node clusters: what about non-wimpy workloads? In DaMoN (2010), pp. 47–55.

[11] MATTSON, R., GECSEI, J., SLUTZ, D., AND TRAIGER, I. Evaluation techniques for storage hierarchies. IBM Systems Journal 9, 2 (1970), 78 –117.

[12] WIGGINS, A., AND LANGSTON, J. Enhancing the scalability of memcached. In Intel Software Network (2012).

[13] Kevin Lim, David Meisner, Ali G. Saidi, Parthasarathy Ranganathan, and Thomas F. Wenisch. 2013. Thin servers with smart pipes: designing SoC accelerators for memcached. InProceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13). ACM, New York, NY, USA, 36-47.