Hardware Acceleration of Key-Value Stores

Sagar Karandikar, Howard Mao, Albert Ou, Yunsup Lee   Advisor: Krste Asanović

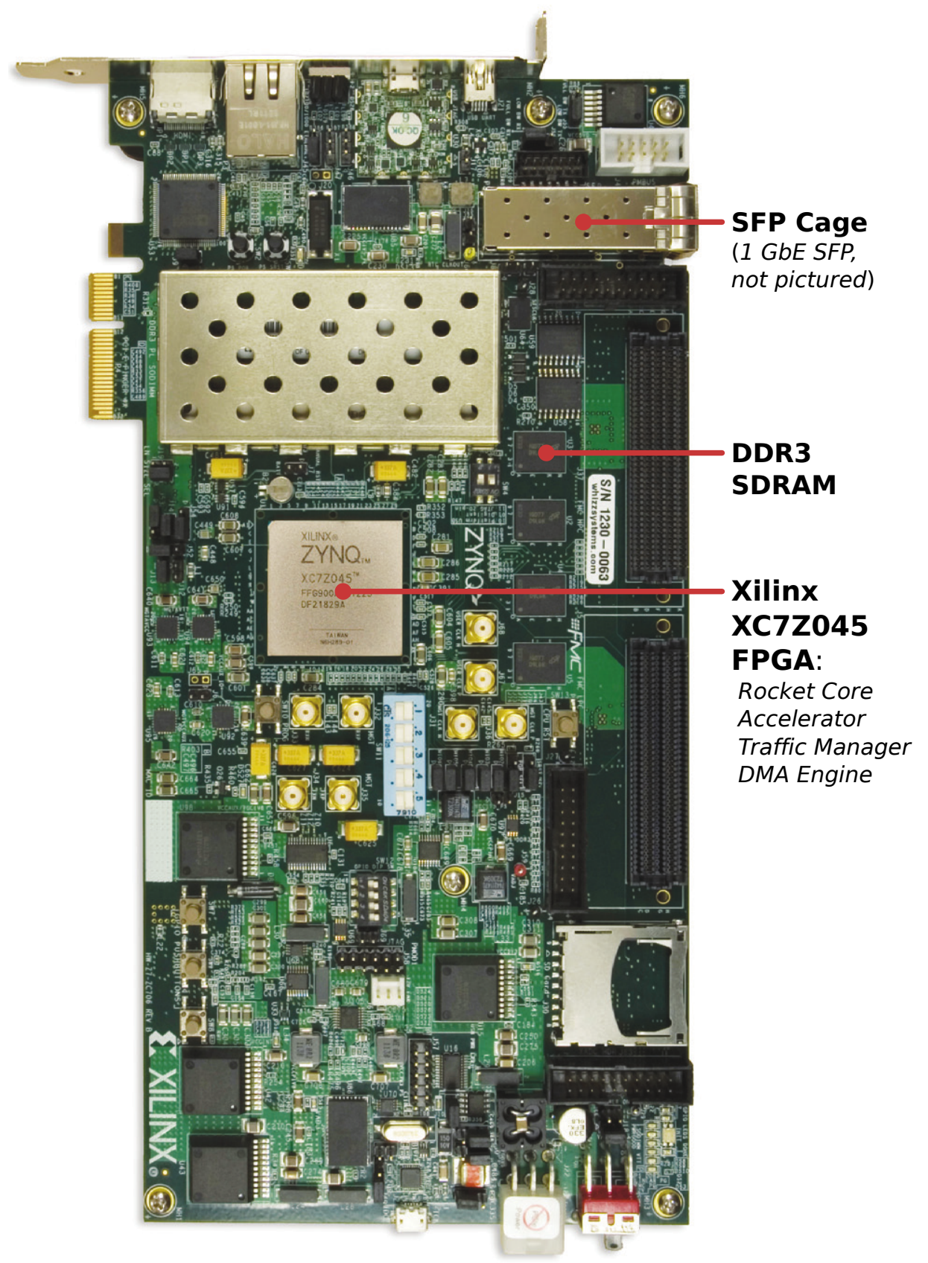
Motivation

- In datacenter applications, path through CPU/kernel/application accounts for 86% of total request latency
- Goal: Serve popular requests without CPU interruption
- Solution: Hardware key-value store attached to the network interface controller
- Many workloads have an access pattern suitable for a small dedicated cache
  - Per a Facebook study, 10% of keys represent 90% of requests
  - Most values are relatively small in size (1 kB)

Related Work

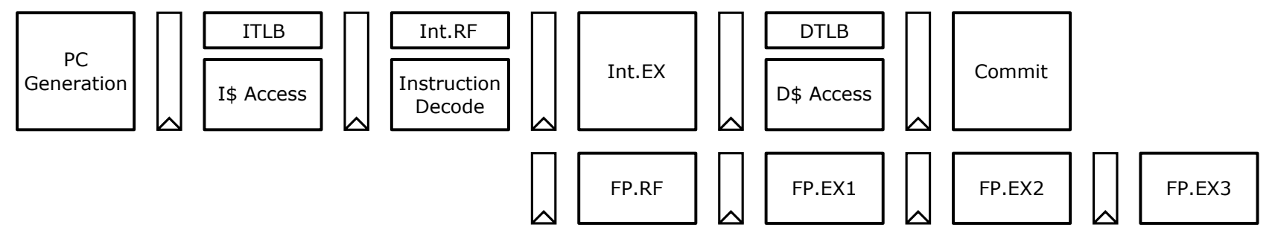
- A 2013 paper by Lim et al. proposed a system dubbed "Thin Servers with Smart Pipes", which served memcached GET requests from FPGA hardware.
- However, the FPGA hardware handled GET requests by accessing DRAM, not a local SRAM cache.

Infrastructure



Xilinx ZC706 Evaluation Platform

- ZYNQ-7000 SoC
- Brocade 1GbE Copper SFP Transceiver
  - Xilinx Tri-Mode Ethernet MAC
  - Xilinx 1000Base-X PCS/PMA
- 64-bit RISC-V Rocket Core (50 MHz)
  - Single-issue, in-order, 6-stage pipeline
  - ASIC version most nearly comparable with ARM Cortex-A5

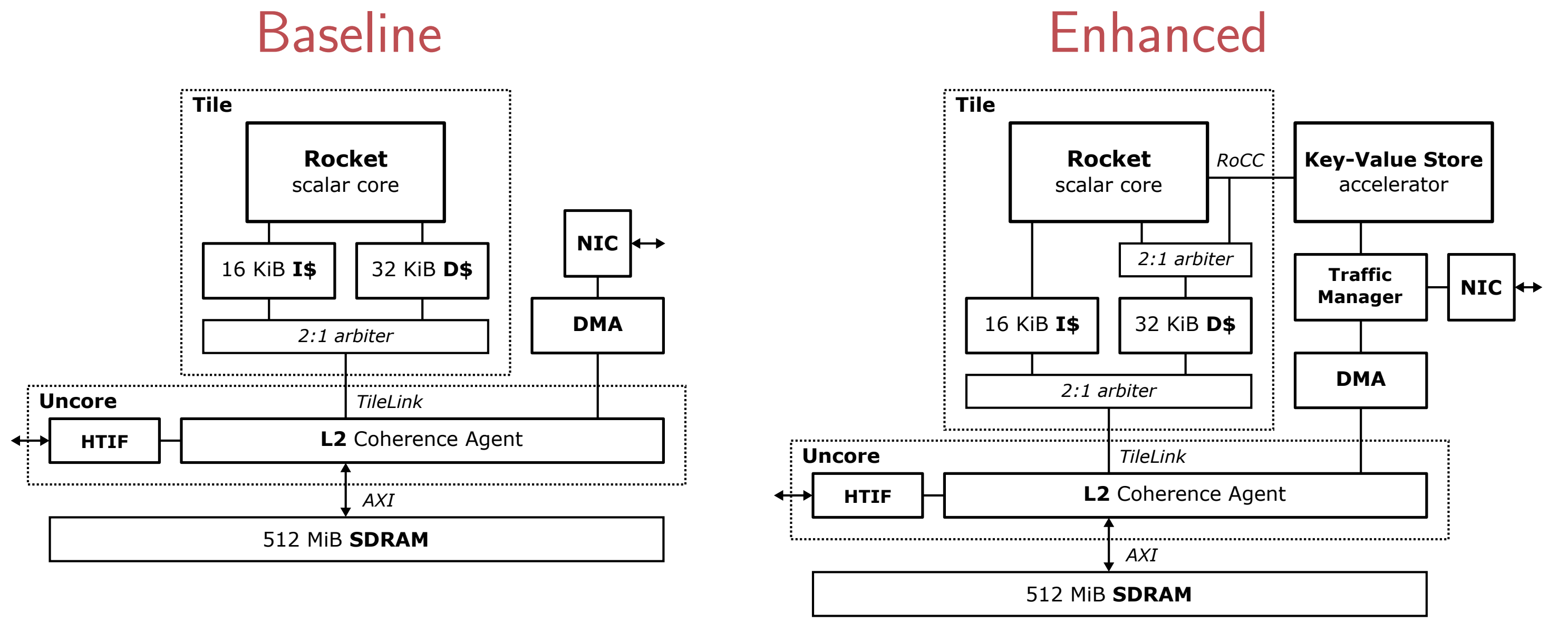


- No pre-existing I/O peripherals for the Rocket core
- Built first RISC-V hardware device: register-mapped NIC
  - Programmed I/O with custom Linux kernel driver
  - First `telnet`/`ssh` session into a physical RISC-V machine
- Evolved to DMA-based NIC for performance

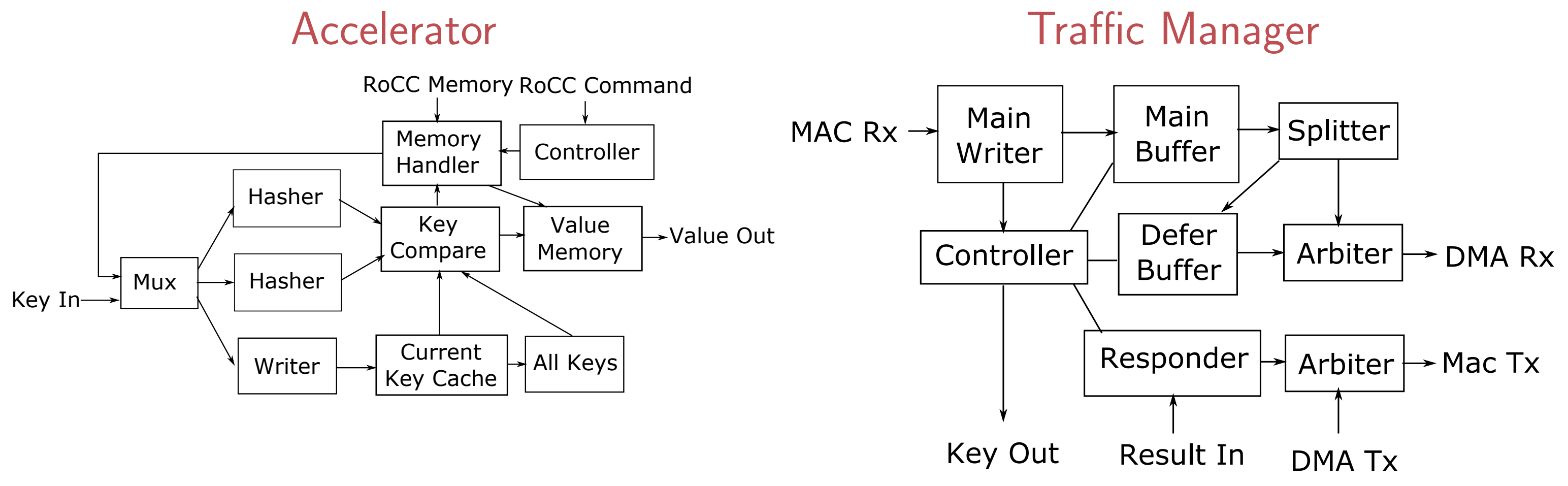
Software

- Manages what keys and values are stored on the accelerator
- Controls the accelerator through the RoCC co-processor interface, which provides custom instructions for setting keys and values
- Responsible for implementing cache replacement policies
  - Identification of the most popular keys as candidates for offloading
  - Invalidation of stale entries

System Architecture



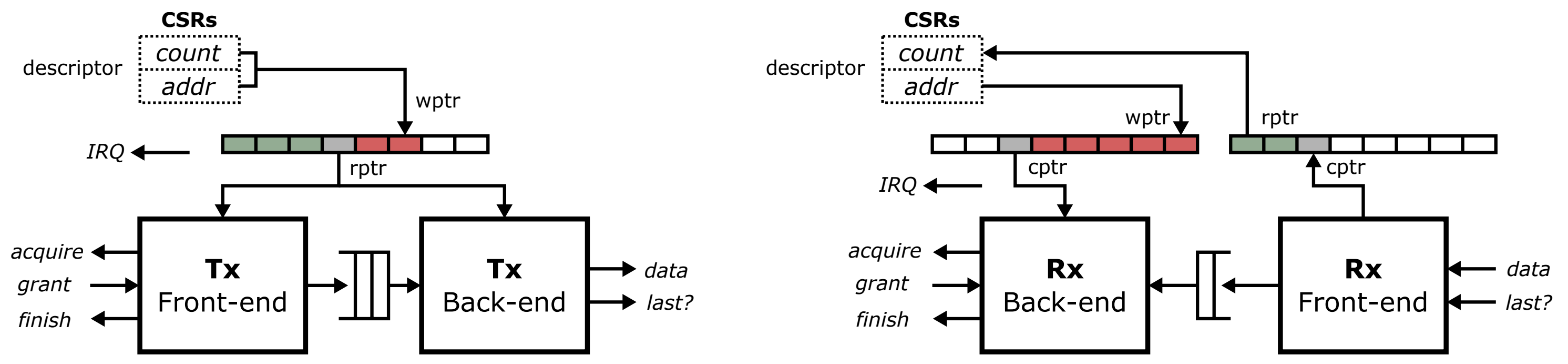
Accelerator



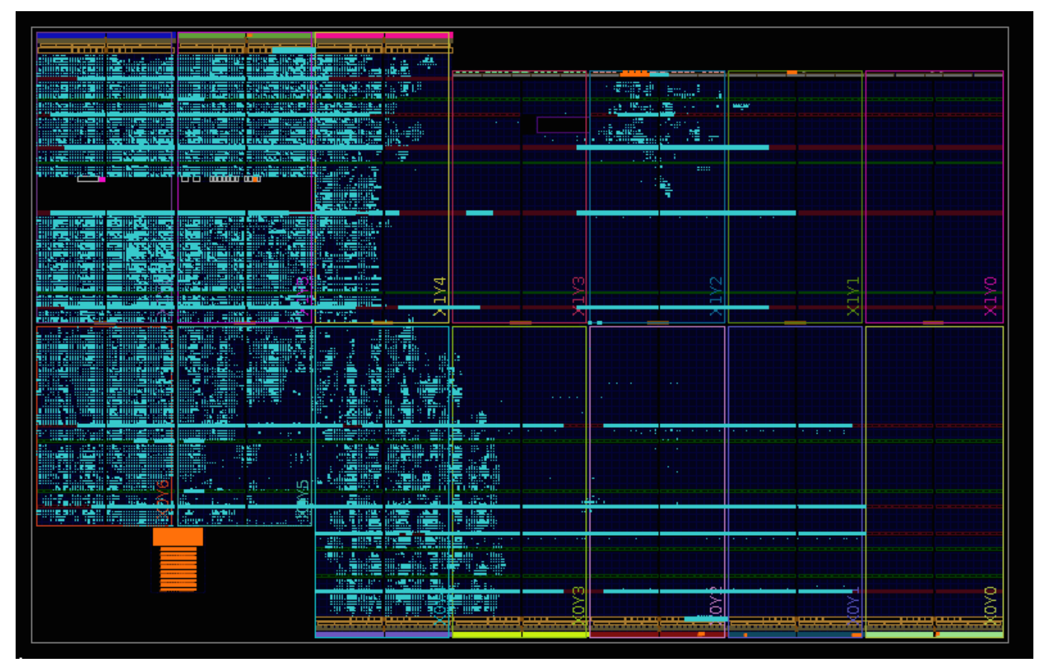
- The accelerator accepts a key and computes primary and secondary hash values, which it uses to retrieve the value from its local block RAM.
- The traffic manager, interposed between the NIC and the DMA engine, implements the specialized Memcached logic.
- For intercepted Memcached GET requests, the traffic manager queries the accelerator and constructs the response packet if the key is found.
- Unhandled frames are forwarded to the DMA engine for processing by the operating system.

DMA Engine

- Performs uncached memory accesses via the TileLink protocol
- Transfers a 512 bit cache block per request
- Front-end/back-end decoupling allows load prefetching to hide memory latency
- Buffer descriptor rings exposed as queues through processor control registers
- Provides 250 times the query throughput as compared to programmed I/O



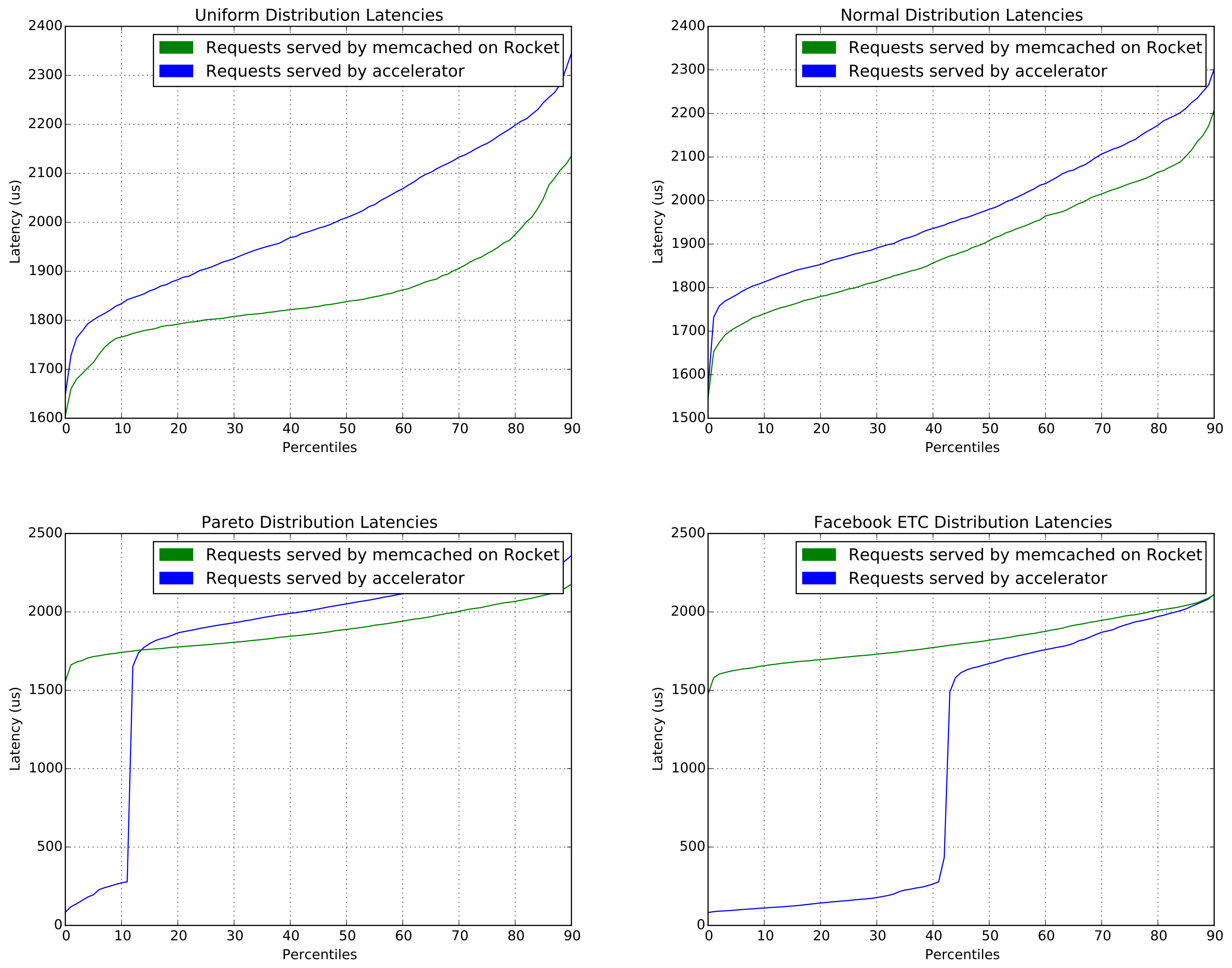
Floorplan



Utilization

Resource	w/o A+TM	w/A+TM
Slice LUTs	17.09%	21.79%
Slice Registers	6.18%	8.01%
Memory	21.65%	63.85%

Latency Evaluation



Conclusion

- By moving some of the keys to the accelerator and serving directly to the NIC from hardware, we gained an order of magnitude speed-up over memcached software running on the Rocket core (1700μs vs 150μs response latency).
- The accelerator serves 40% of keys at this reduced latency for Facebook ETC.
- However, we still have a long way to go before reaching production quality.

Future Work

- Place the DMA engine, traffic manager, and accelerator in faster clock domains with asynchronous FIFOs, rather than be constrained by the core frequency
- Widen I/O interfaces for greater throughput
- Investigate replacing the fixed-function traffic manager with a programmable co-processor (reminiscent of mainframe channel I/O)
- Conduct torture testing for reliability
- Explore opportunities for measuring and improving energy efficiency

University of California, Berkeley   ASPIRE Winter 2015 Retreat   Department of Electrical Engineering & Computer Sciences