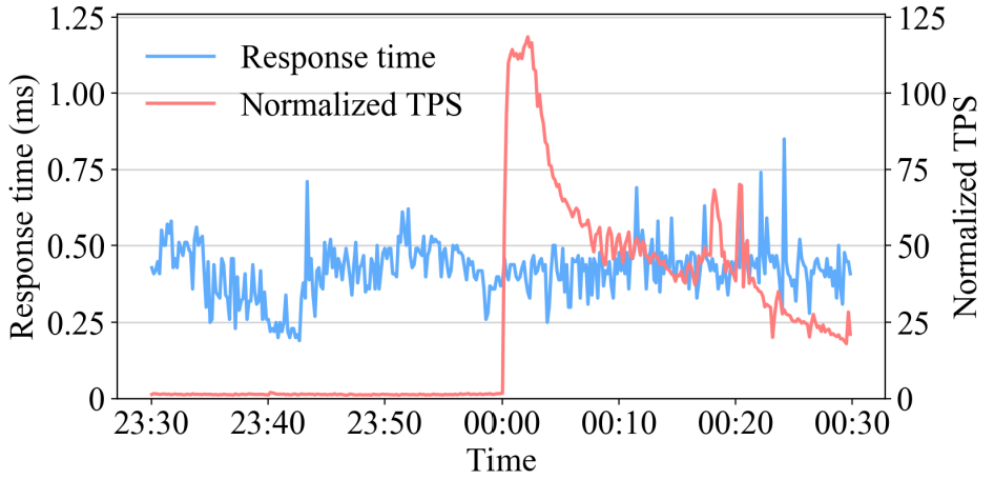
FPGA-Accelerated Compactions for LSM-based Key-Value Store

Teng Zhang, Jianying Wang, Xuntao Cheng, Hao Xu, Nanlong Yu, Gui Huang, Tieying Zhang, Dengcheng He, Feifei Li, Wei Cao, Zhongdong Huang, Jianling Sun

Key-Value Store in the Cloud

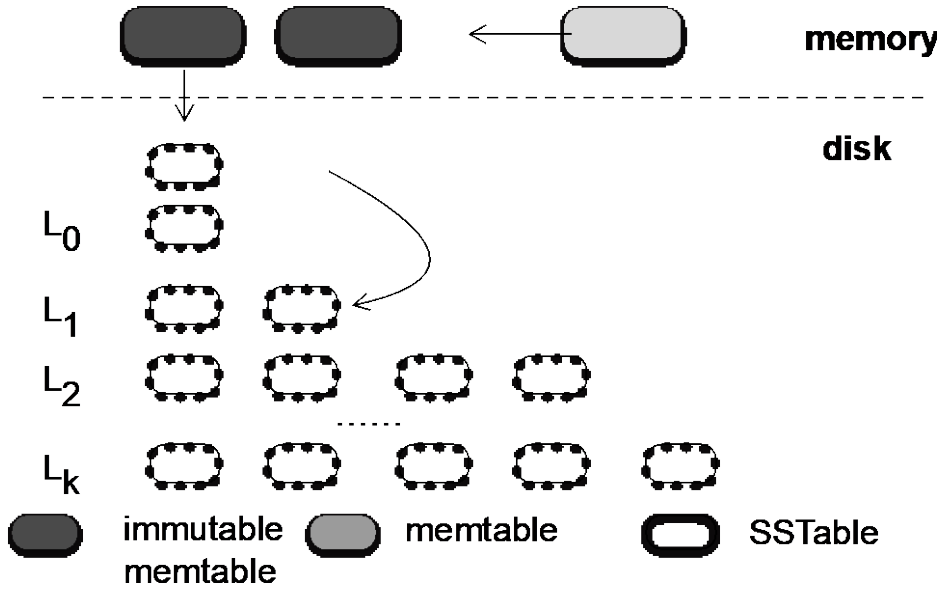
- WPI Workloads [SIGMOD'19]
  - **544 K** sales transactions per second (11 Nov, 2019)



- TCO (Total cost of ownership)
  - Money burning SSDs
  - Power Consumption

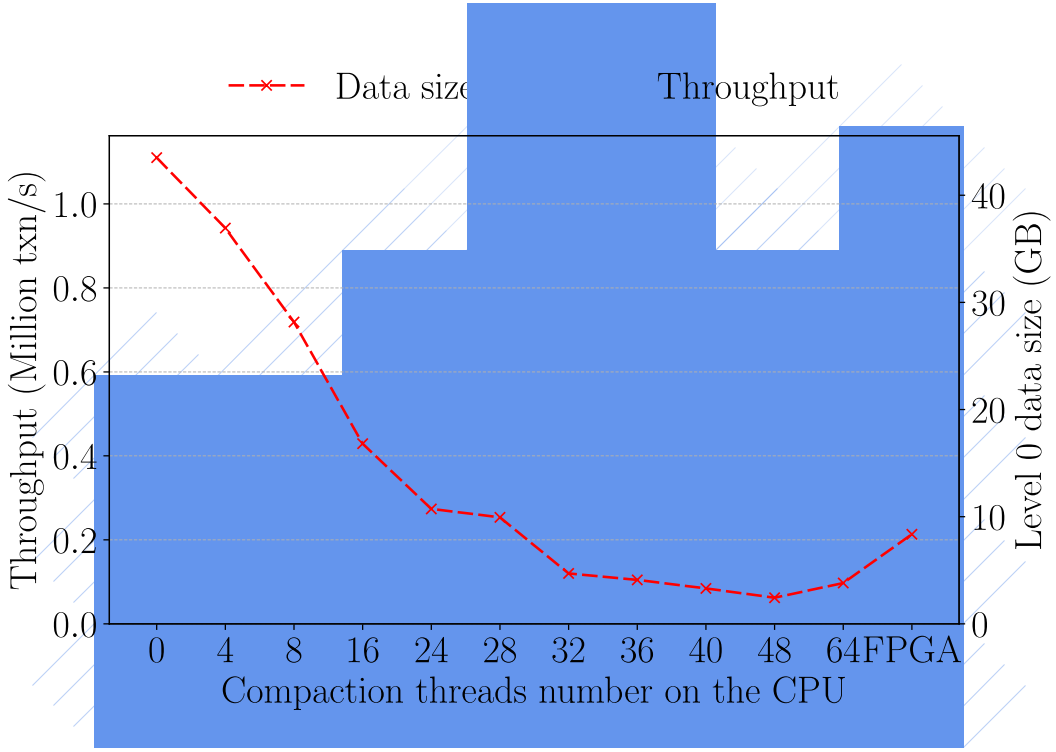
LSM-based Key-Value Store

- Buffer Changes in memory
- Tiered Storage
- Compaction
  - Merge KV records to shorten read path
  - Reclaim storage space



The Performance Fatigue Problem

- Shattered L_0
  - Overlapped key ranges
  - Single lookup may incur multiple I/Os



The Performance Fatigue Problem

- Shifting Bottlenecks
  - CPU-bounded for short KV records
  - IO-bounded for long KV records
  - Increasing I/O ability

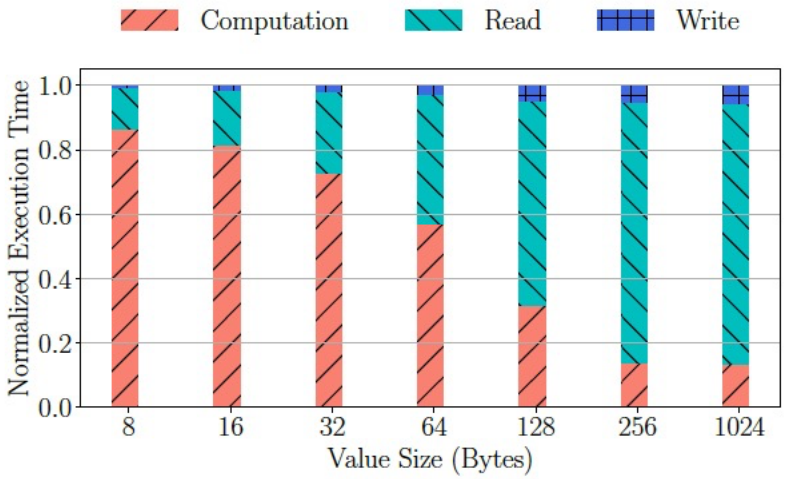
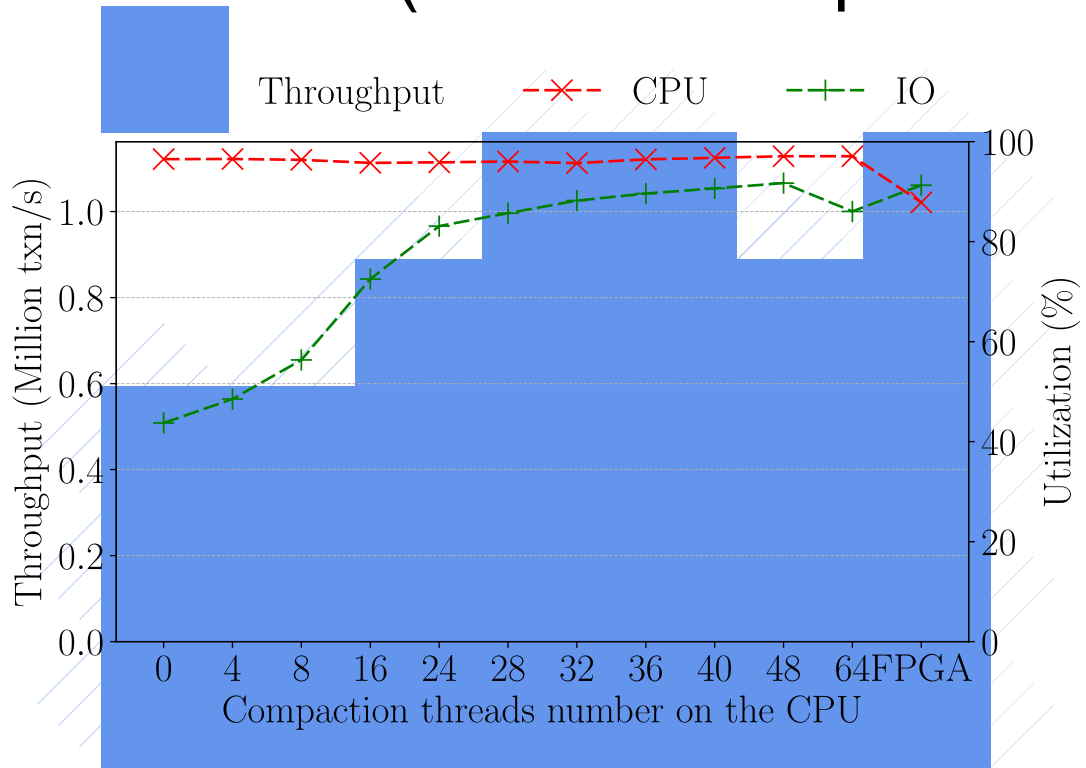


Figure 3: The breakdown of CPU compaction (key=8 bytes).

The Performance Fatigue Problem

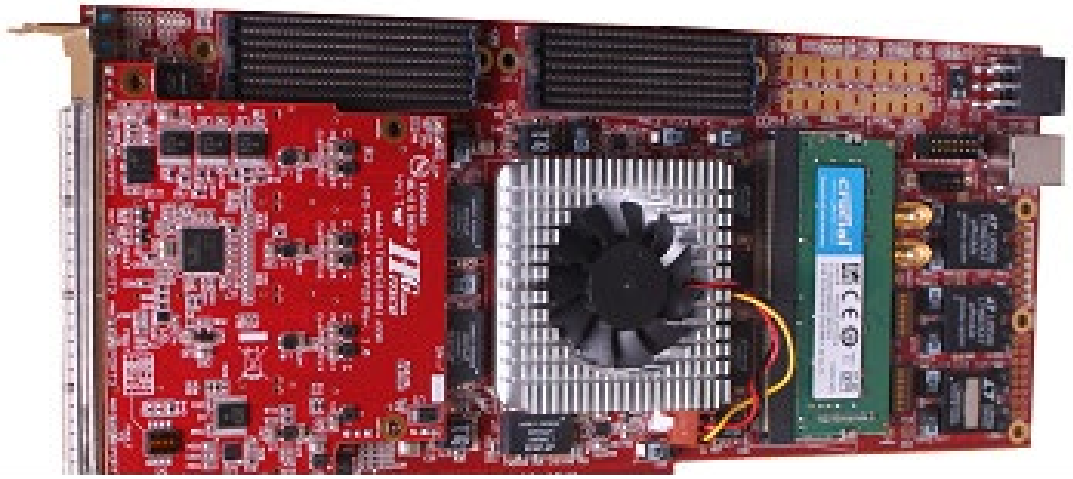
- K = 8B, V=32B, DBBench R:W = 3:1
- Shattered L_0 (0 ~ 32 compaction threads)
- CPU Resource contention (32 ~ 64 compaction threads)



Offloading Compactions to FPGAs

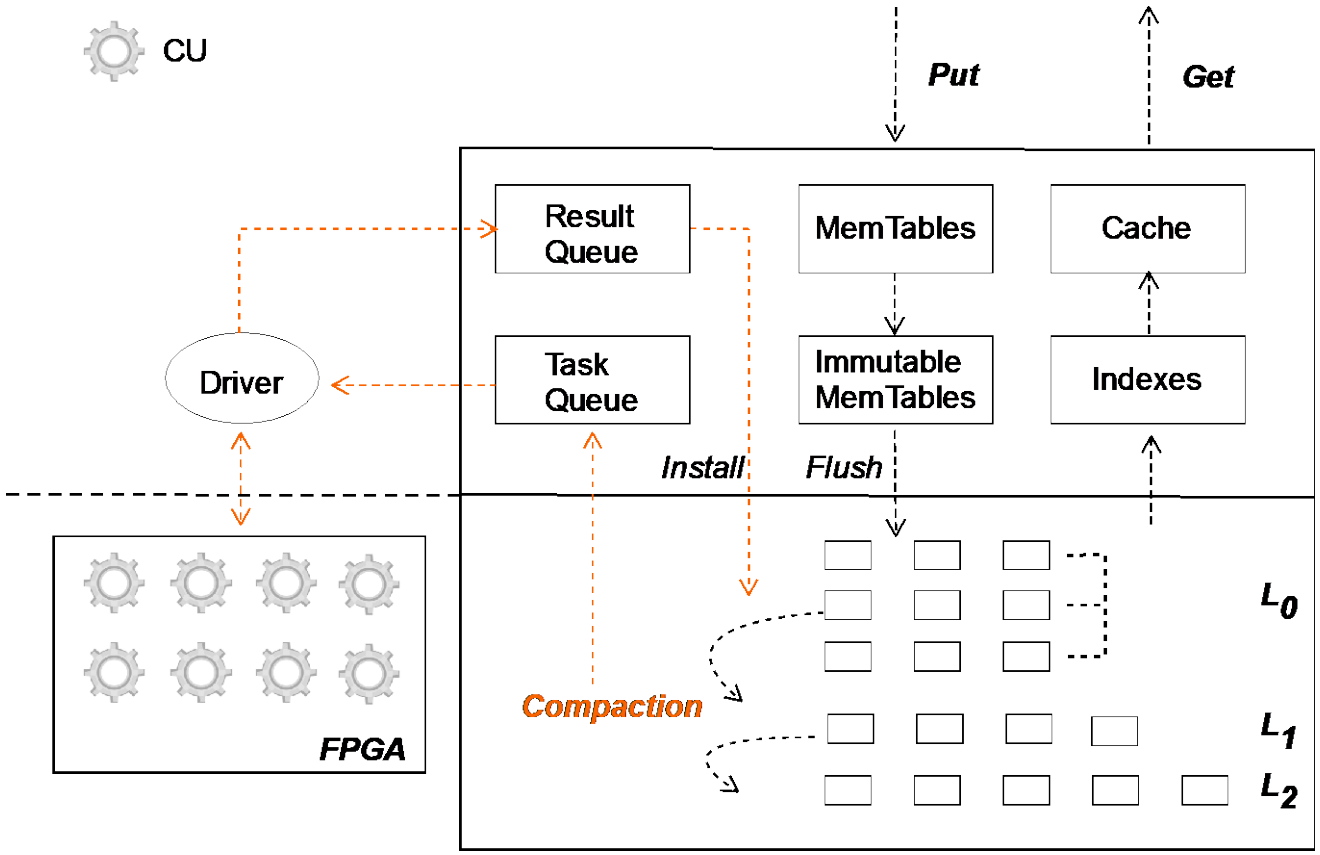
- Compaction can be pipelined

- Lower power consumption



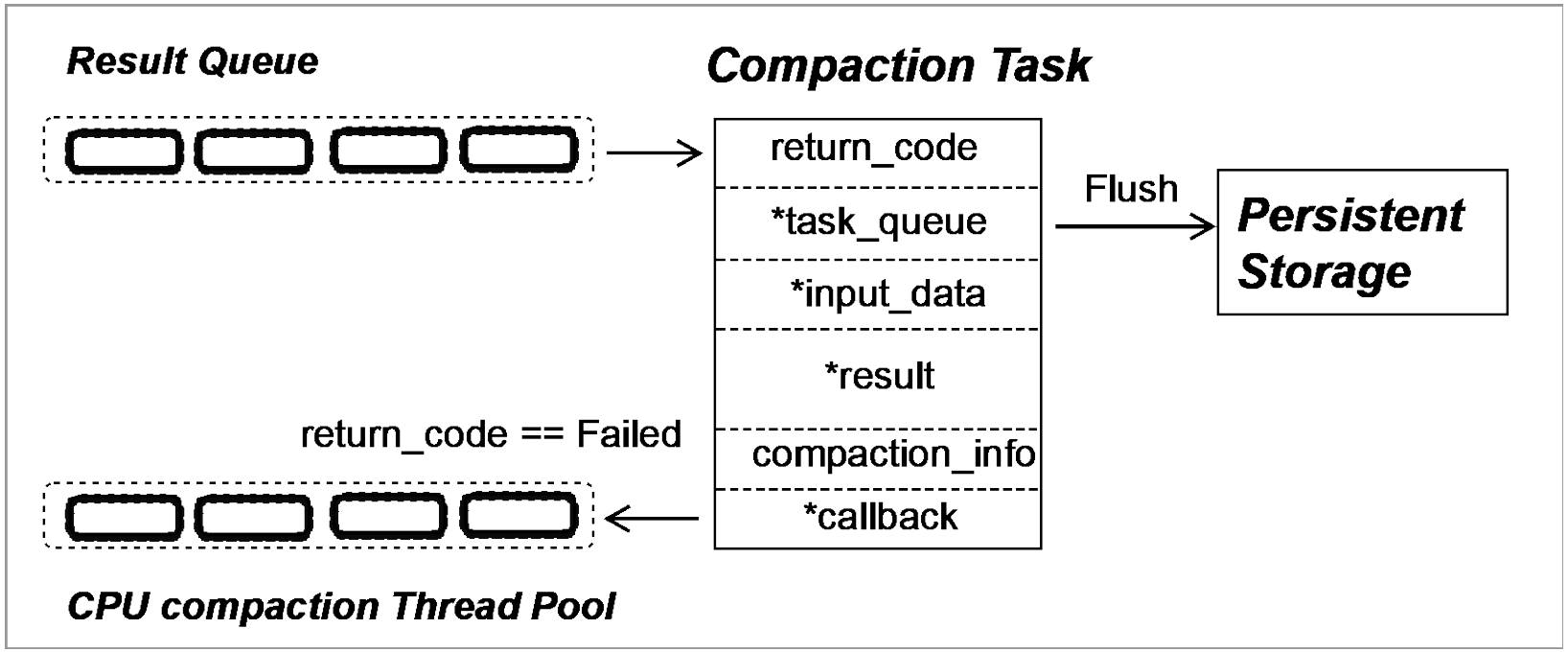
Integrating FPGA as a co-processor

- X-Engine [SIGMOD19]
- Result/Task Queue
- Compaction Units
- Driver



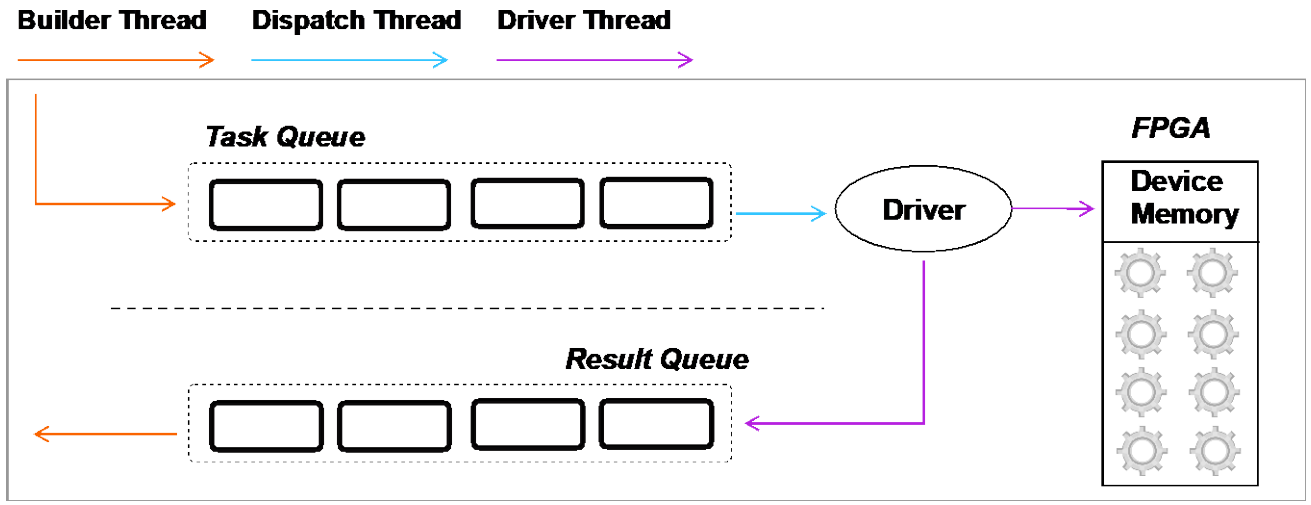
Integrating FPGA as a co-processor

- Compaction Task Structure



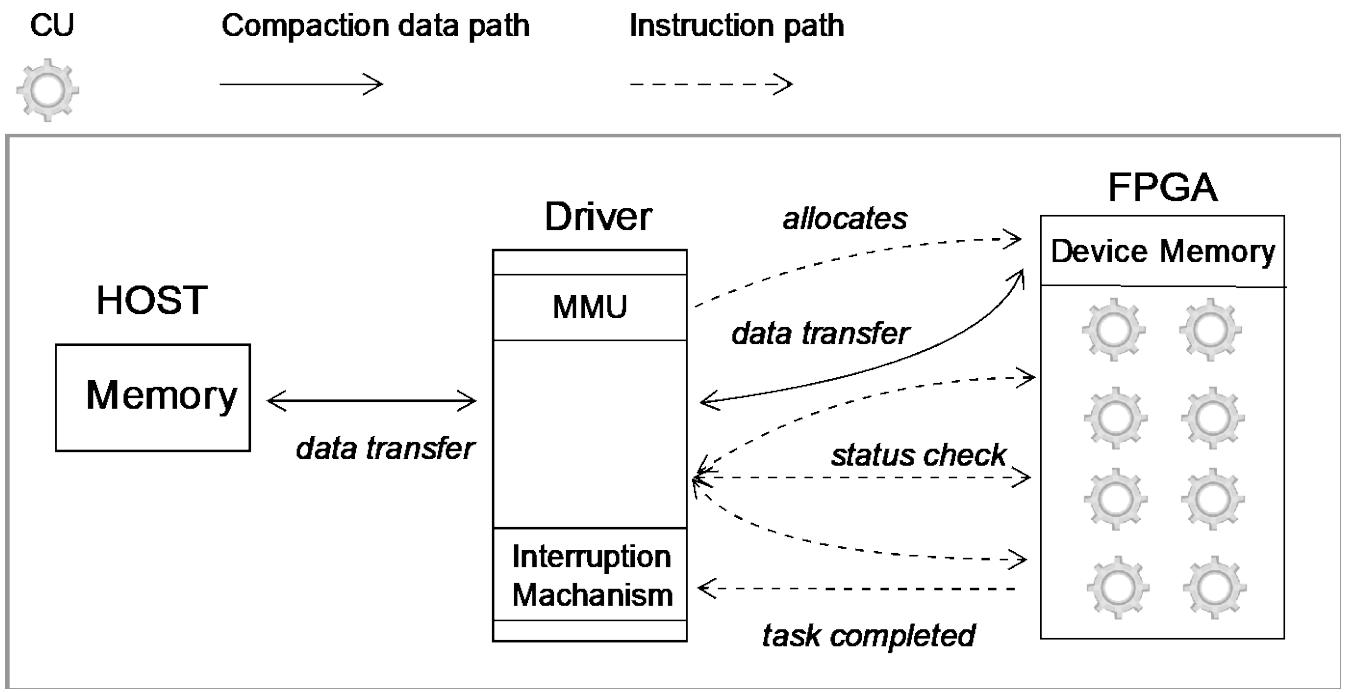
Integrating FPGA as a co-processor

- Compaction Task Management
  - Builder Thread (construct tasks by partitioning extents into similar size)
  - Dispatch Thread (round-robin)
  - Driver Thread (transfer data to the device memory, notify CU to work)



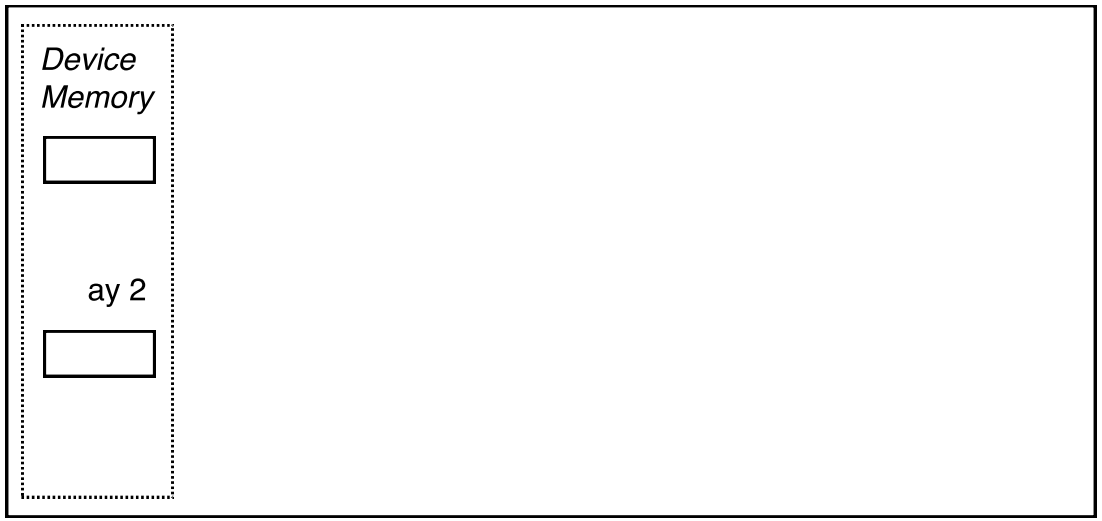
Integrating FPGA as a co-processor

- Separate Path for Data and Instructions Transfer



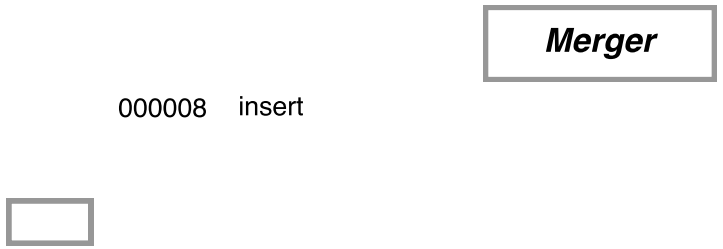
Compaction Unit

- 4-way Decoder
- Encoder
- KV Ring Buffer
- Controller to Coordinate the Pace of Each Module



Compaction Unit

- KV Transfer, Key Buffer and Merger



Compaction Unit

- Analytical Model
  - Workload-dependent Merging selectivity

$$T_{CU} = min\{T_{decoder}, T_{kv\_transfer}, T_{cpe}, T_{encoder}, T_{mem}\}$$

$$T_{decoder} = \frac{f_{FPGA}}{b_{decoder} + (A_{decoder} \cdot W_{key} + W_{value})/W_{bus}}$$

$$T_{encoder} = \frac{f_{FPGA}}{b_{encoder} + (A_{encoder} \cdot W_{key} + W_{value})/W_{bus}}$$

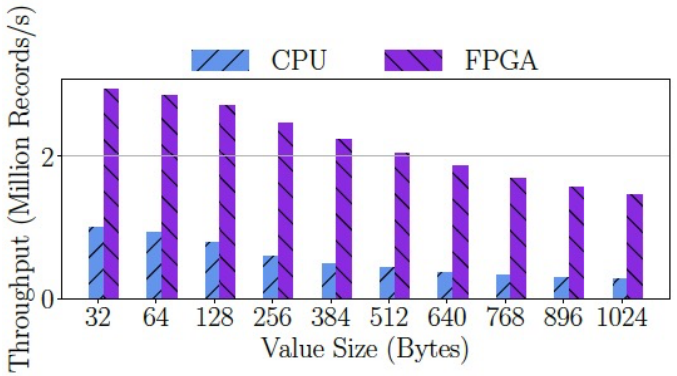
Parameter	Values/Units	Description
N	Workload-dependent	Total number KVs processed
f_{FPGA}	200 MHz	Clock frequency of the FPGA
W_{bus}	8 Bytes	Width of bus data
W_{key}	$1 \sim 2$K Bytes	Width of key
W_{value}	$0 \sim 4$K Bytes	Width of value
$A_{decoder}$	2	Decoder amplification factor[1]
$A_{kv\_transfer}$	1	KV_transfer amplification factor
A_{merger}	5	CPE amplification factor
$A_{encoder}$	2	Encoder amplification factor
$B_{transfer}$	Bytes/second	Host-device transfer bandwidth
$b_{decoder}$	10	Base cycles for Decoder
$b_{kv\_transfer}$	18	Base cycles for KV_transfer
b_{merger}	50	Base cycles for Compaction PE
$b_{encoder}$	46	Base cycles for Encoder
μ	Workload-dependent	Merging selectivity

Experimental Setup

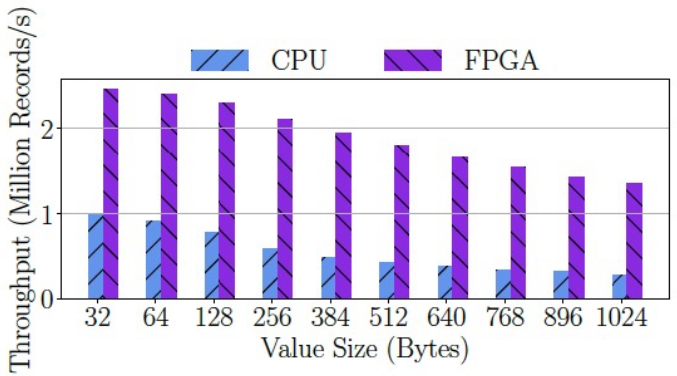
- Two Intel Xeon Platinum 8163 2.5 GHz 24-core CPUs with two-way hyperthreading
- 768 GB Samsung DDR4-2666 main memory
- RAID 0 consisting of 10 Samsung SSDs
- Xilinx VU9P FPGA (200MHZ, 16GB device memory)

Evaluating the FPGA-based Compaction

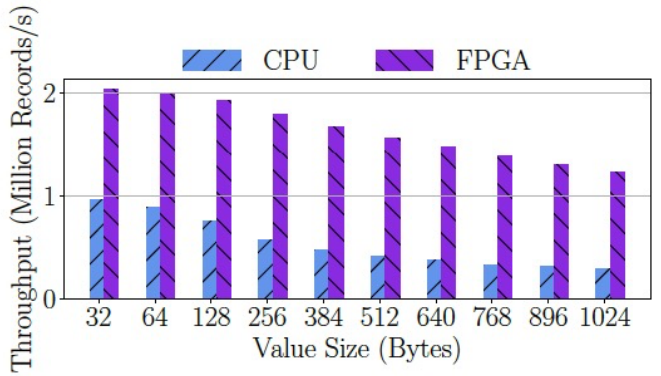
- 2~5x Speedup Comparing to a Single CPU Thread



(a) 8-byte key.



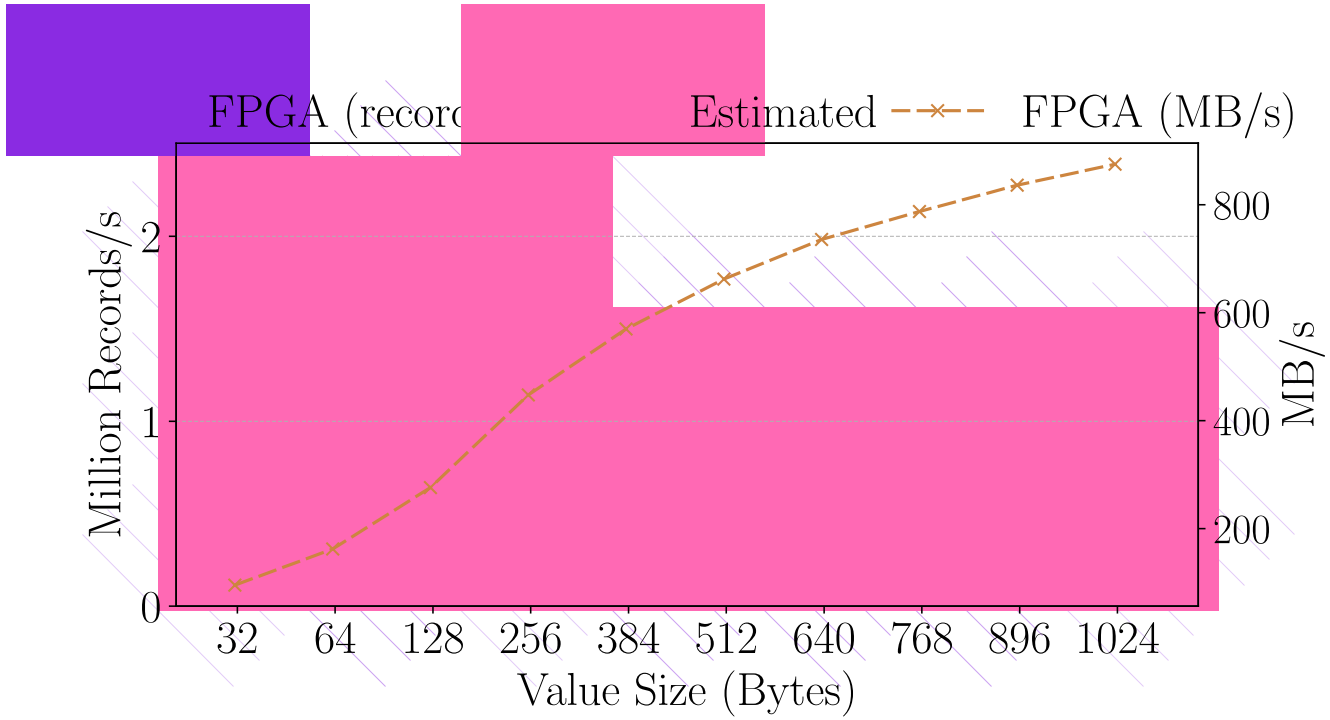
(b) 32-byte key.



(c) 64-byte key.

Analytical Model Validation

- Within 5% error for short KVs, 13% mismatch for 1024B value
- Potential Pipeline Stalls and Bus Contention



Compaction Unit Resource Consumption

- In Current Design, Up to 8 CUs Placed in the FPGA Board
- 50% Utilization of the FPGA Resource

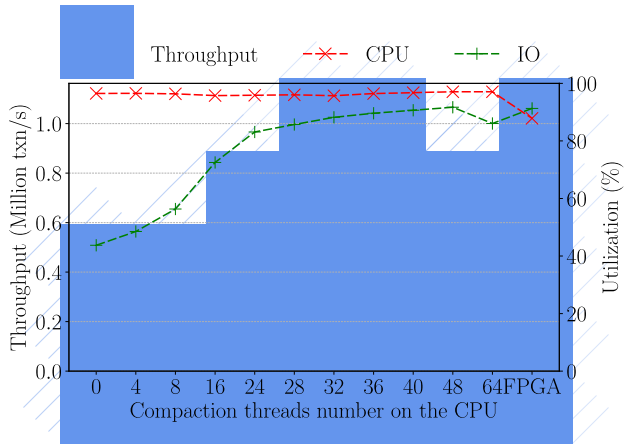
	LUT	**Flip-Flop**	**RAM (MB)**
Decoder	5783 × 4	4570 × 4	8 × 4
KV Transfer	1076	1006	0
Merger	4119	2555	0
Encoder	6489	4101	0
Others	3819	4841	14
1 CU	38635	30783	46
FPGA total	1182000	2364000	960
Utilization	**3.2%**	**1.3%**	**4.8%**

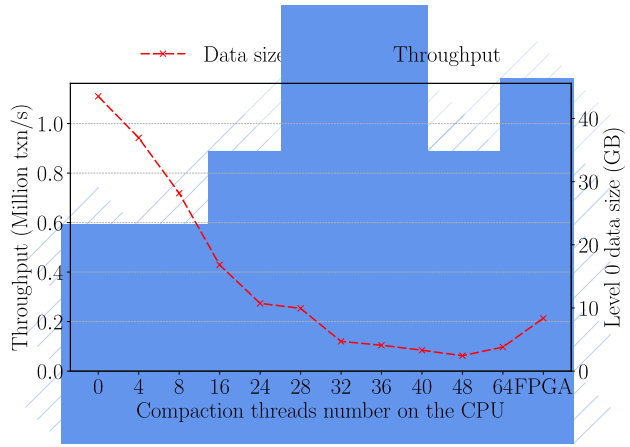
Evaluating a KV Store with FPGA offloading of Compactions

- Typical WPI Workloads
  - R:W = 3:1
  - Ker = 8B, Value = 32B (Common size for secondary indexes)
- Preload 32 LSM-tree tables, each storing 200 million records
- 3600s Measurement after 3600s Warm-up

The Value of Adding an FPGA

- Improve Throughput by **23%**
- Increase Efficiency (Transactions per Watt) by **31.7%**

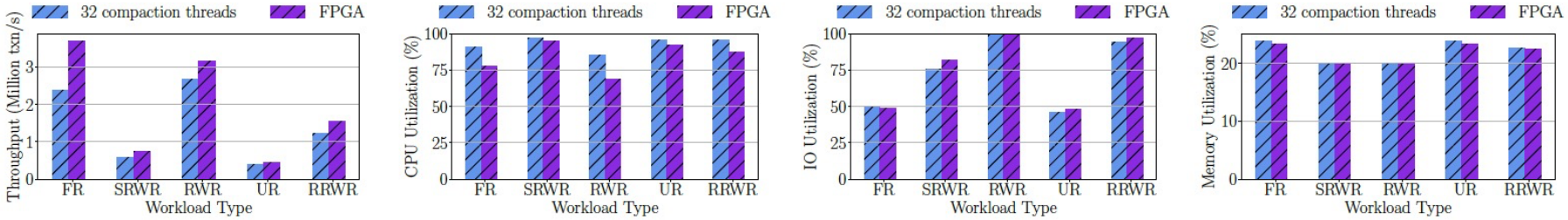




	Million Txn/s	*Avg Get RT (µs)*	*P99 Get RT (µs)*	*Avg Put RT (µs)*	*P99 Put RT (µs)*	*Power (Watt)*	*Efficiency (Txn/Watt)*
CPU (32 compaction threads)	0.90	139.89	928.15	107.51	864.95	636.54	1416.54
CPU + FPGA	1.11	119.38	537.08	77.30	499.52	590.78	1865.44
Improvement	**+23.3%**	**-14.7%**	**-42.1%**	**-28.1%**	**-42.2%**	**-7.2%**	**+31.7%**

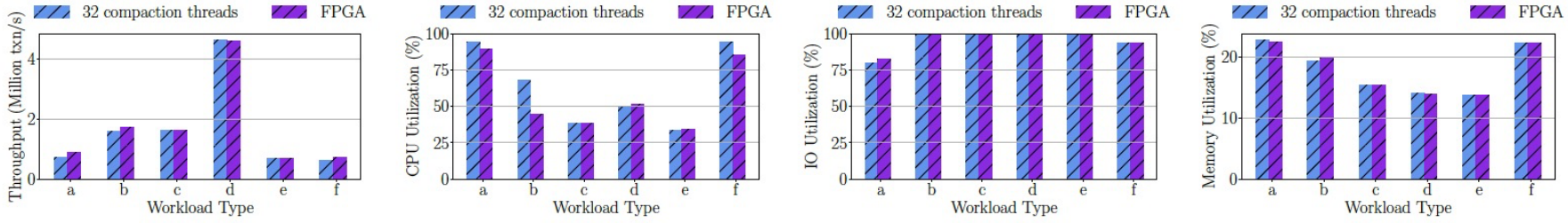
Macro-benchmark

- DBBench



(a) Throughput. (b) CPU Utilization. (c) I/O Utilization. (d) Memory Consumption.

- YCSB



(a) Throughput. (b) CPU Utilization. (c) I/O Utilization. (d) Memory Consumption.

Evaluation Takeaways

- In-time Compaction is vital to LSM-based KVS.
- FPGA offloading solution achieves lower power consumption and yields better efficiency.
- FPGA Acceleration helps most in WPI Workloads and short KVs.

Conclusion

- We identify the bottleneck of LSM-based KVS caused by slow and heavy compactions.
- We design an efficient compaction pipeline on FPGAs and integrate it with X-Engine.
- We model the FPGA compaction within 13% error.
- The FPGA offloading solution increase the efficiency by 31.7%.

Q & A