Reinforcement Learning-Based SLC Cache Technique for Enhancing SSD Write Performance

Sangjin Yoo and Dongkun Shin
Sungkyunkwan University, Korea
newlandlord@skku.edu, dongkun@skku.edu

Qual-level-cell (QLC) flash memory

- A mainstream storage medium of solid-state drives (SSDs)
- Higher density and lower cost
- Slower performance and lower endurance
  - especially, significantly worse write performance

	SLC	TLC	QLC
Program time (page)	160 us	730 us	3102 us
Read time (page)	30 us	66 us	140 us
Erase time (block)	3 ms	4.8 ms	3.5 ms
Endurance (Max. P/E)	100,000	3,000	1,000

[Comparison of SLC, TLC and QLC flash memory][1]

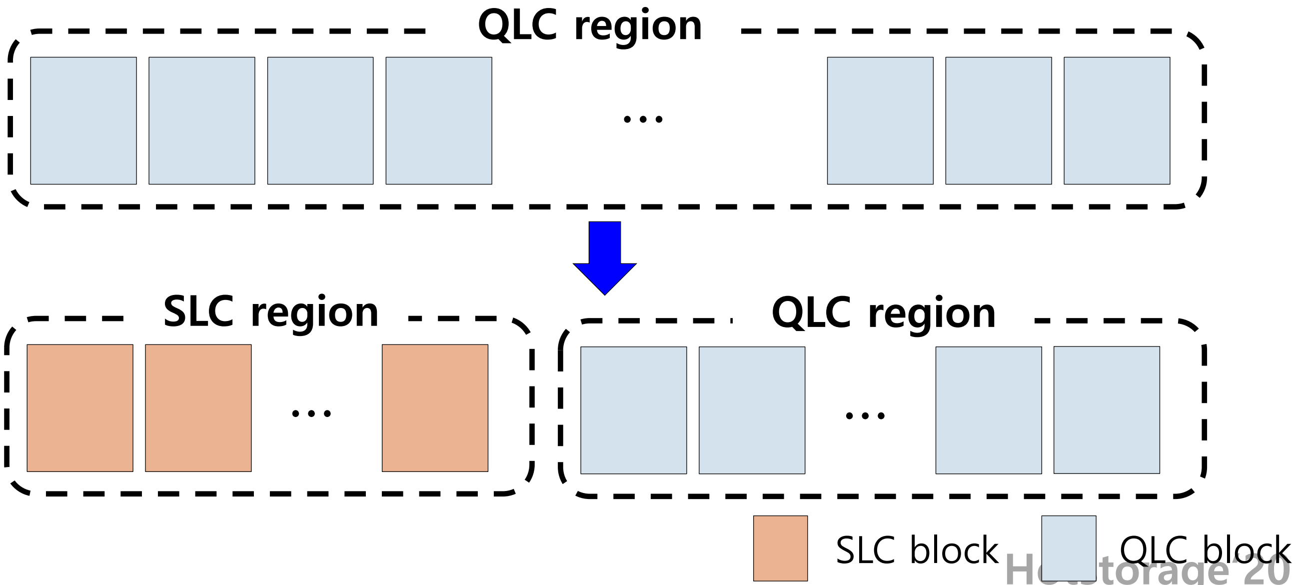
[1] Analysis on Heterogeneous SSD Configuration with Quadruple-Level Cell NAND Flash Memory, 2019

Hybrid SSD Architecture

- A partitioned SLC region
  - a cache space of the remaining QLC region
  - hide the slow performance of QLC flash memory

Typical SSD Architecture

Hybrid SSD Architecture

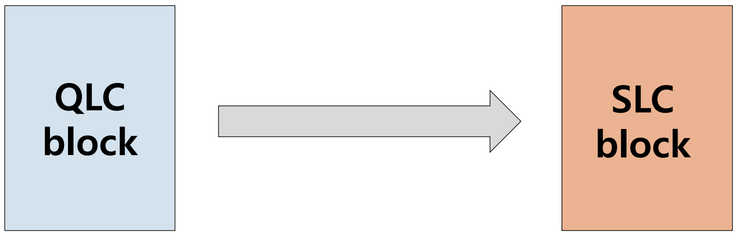


Important factors in the hybrid SSD

1. SLC region size

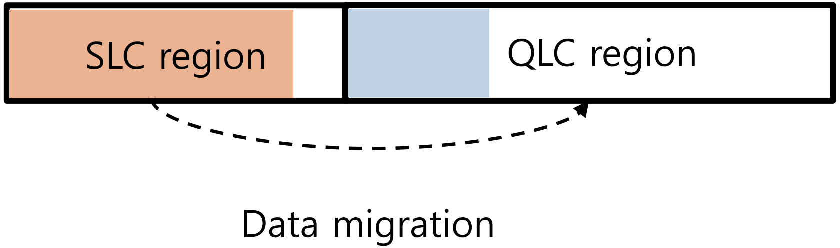
 - considering the trade-off between capacity loss and SLC-to-QLC migration overhead

Capacity loss



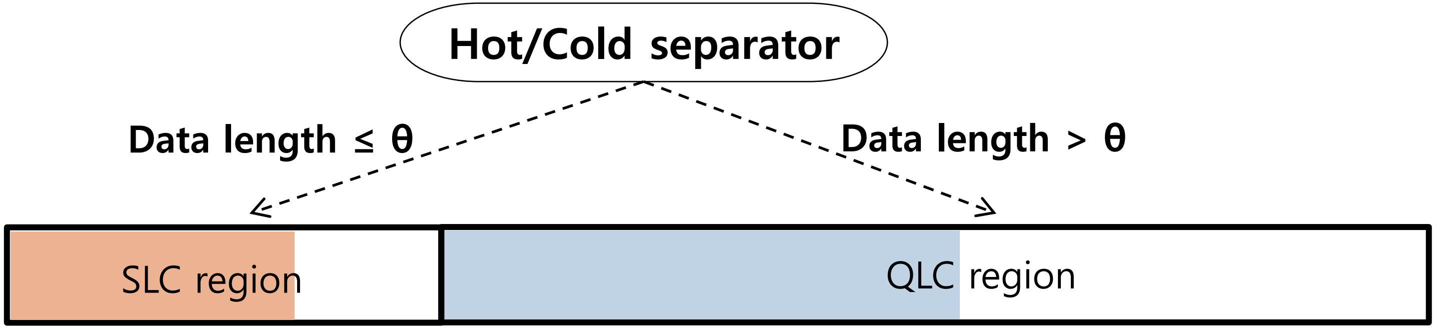
***Capacity (SLC block) = Capacity (QLC block) / 4**

SLC-to-QLC migration



Important factors in the hybrid SSD

2. Hot/cold separation threshold
   - write only frequently-updated (hot data) at SLC region
   - small data tend to be frequently updated[2]
     - write request size can be used to distinguish between hot data and cold data



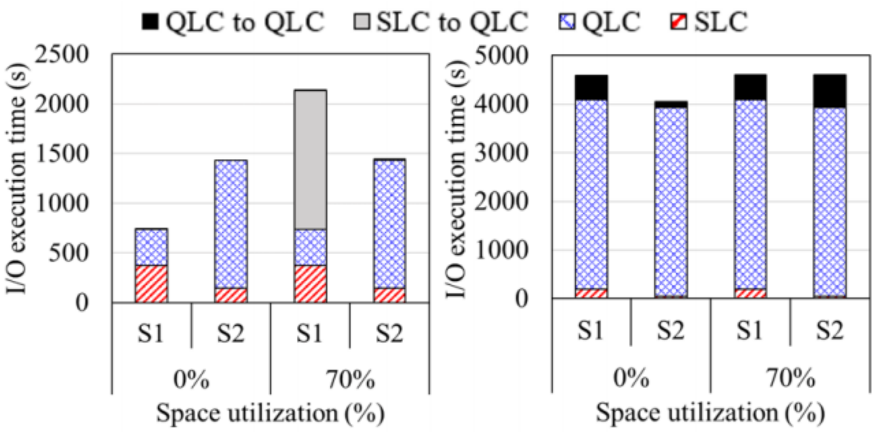
[2] LAST: locally-aware sector translation for NAND flash memory-based storage system, 2008

SLC cache management schemes

- Two types of hybrid SSDs
  - Static scheme
    - fixed SLC cache size and fixed hot/cold separation threshold
  - Dynamic scheme
    - adjust the SLC region parameters depending on the system states (e.g., amount of stored data, I/O access pattern, etc.)
- Recent QLC SSDs adopt the dynamic scheme-based hybrid SSD architecture
  - The proper SLC cache sizes at different space utilizations are investigated at offline with representative workloads
  - Not exact under unexamined or variable workloads

Problem of the current dynamic hybrid SSDs

- Optimal policy is different depending on space utilization and workload



(a) PC (b) YCSB-A

Hot/cold separation threshold : setting1(64KB), setting2(16KB)

Space utilization (%)	0 ~20	20 ~30	30 ~40	40 ~50	50 ~60	60 ~70	70 ~100
Setting 1	56	50	40	30	25	20	10
Setting 2	40	40	30	25	20	10	5

[A table of the SLC cache size]

- Need a more intelligent algorithm
  - to adjust the SLC cache parameters considering the changing system states

Reinforcement Learning for dynamic SLC cache

- Q-learning
  - to learn the optimal SLC cache parameters according to the system states
  - calculates Q-values that tell which action is right in a given state

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a} Q(s',a') - Q(s,a))$$

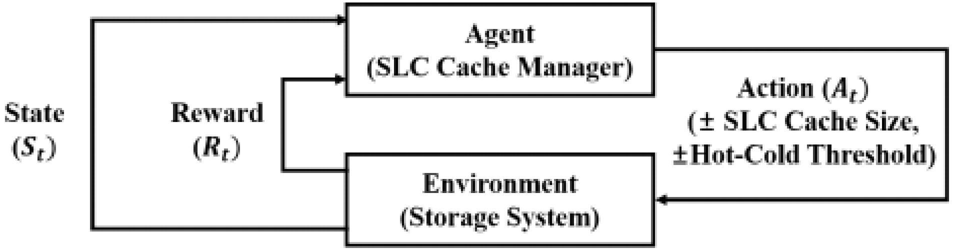
- $a(action), s(state), r(reward), s'(next\ state), a'\ (action\ in\ s'), \alpha(learning\ rate), \gamma(discount\ factor)$

  - size of (Q-table) = # of states x # of actions
  - ε-greedy algorithm
    - Set ε to 0.07 in our experiments

$$\pi(s) = \begin{cases} a^* = argmax_a Q(s,a), 1-\varepsilon \\ a \neq a^*, \varepsilon \end{cases}$$

Reinforcement Learning for dynamic SLC cache

- Environment
  - Defines the state S_t based on the workload characteristics and the internal status of the SSD, and estimates the reward R_t
- SLC cache manager
  - Select an action A_t including changes of the SLC cache size and hot/cold separation threshold



[SLC cache management with RL]

Algorithm 1 SLC Cache Management

Input: State (S_t), State (S_{t-1}), Action (A_{t-1})
Output: Action (A_t)

A_t = GetAction(S_t)
Perform A_t
R_t = GetReward( )
Update Q-value (S_{t-1}, A_{t-1}) with Equation 1

State

- Observe to know the change of environment
  - includes both the host and the SSD subsystem
  - Q-table size = 5,184 bytes (=1,296 state x 4 bytes)

Category	Information used for State	# of bins
SSD	SLC cache size	9
	Space utilization	4
	Previous action	9
Host Workload	Demand for SLC writes	2
	Update write frequency in SLC cache	2

Reward

- Need to consider all write costs to calculate the reward of the previous action
  - SLC/QLC write latency of SLC/QLC mode
  - Delayed time by migration and QLC garbage collection

Algorithm 2 Reward function

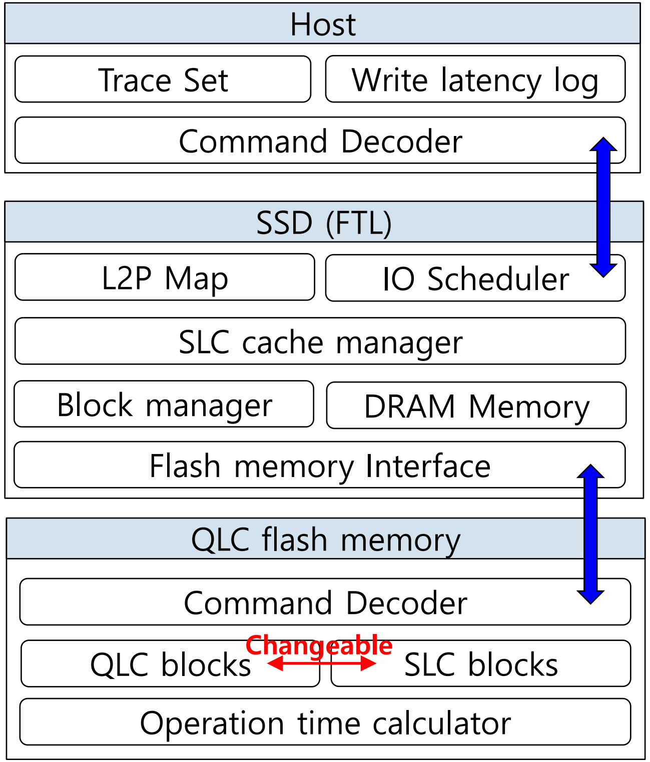
Input: $T_{SLC-to-QLC}$, $T_{QLC-to-QLC}$, $T_{SLCwrite}$, $T_{QLCwrite}$, space utilization U

Output: Reward (R_t)

reclaim cost = $T_{SLC-to-QLC} + T_{QLC-to-QLC}$
host write cost = $T_{SLCwrite} + T_{QLCwrite}$
total write cost = (1-U)×host write cost + U×reclaim cost
if total write cost > average total cost **then**
    R_t = negative reward
else
    R_t = positive reward
end if
Update average total write cost

Experiments

- QLC-based Hybrid SSD Simulator
  - 32GB density (1channel, 1bank)
  - Total 2,138 blocks + over-provision 3%
  - 256 pages/SLC block, 1024 page/QLC block
  - Page size : 16KB
  - DRAM memory : 144KB
- FTL
  - 4KB Page-level L2P mapping
    - Fully cached address mapping table
  - GC or migration trigger condition
    - # of free block of each region ≤ 5



[Our trace-driven simulator]

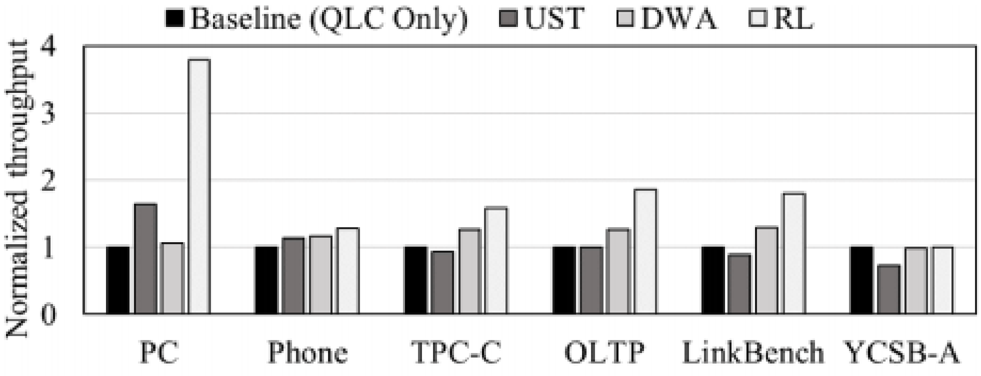
Experiments

- Compared with two previous dynamic SLC techniques
  - Utilization-aware self tuning (UST)[3]
  - Dynamic write accelerator (DWA)[4]
  - Baseline: use only QLC blocks without SLC cache
- Workload characteristics

Trace		PC	Phone	TPC-C	OLTP	LinkBench	YCSB-A
Address space (MB)		1,029	7,606	4,622	5,694	4,482	30,241
Total write amount (MB)		46,426	81,833	39,506	25,866	38,391	97,294
Avg. request size (KB)		66.7	42.8	34.3	35.8	28.2	896.3
Write request size distribution (%)	≤ 128KB	29.47	25.22	53.25	53.4	61.76	0.09
	≤ 256KB	23.08	1.47	2.06	5.01	3.21	0.06
	≤ 512KB	27.03	1.76	16.05	12.42	15.58	3.87
	> 512KB	20.42	71.55	28.64	29.17	19.45	95.98

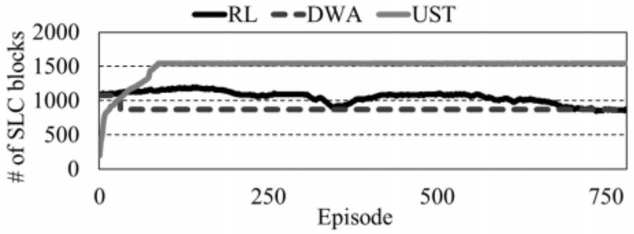
[3] Utilization-aware self-tuning design for TLC flash storage devices, 2016
[4] Optimized client computing with dynamic write acceleration, 2014

Write Throughput

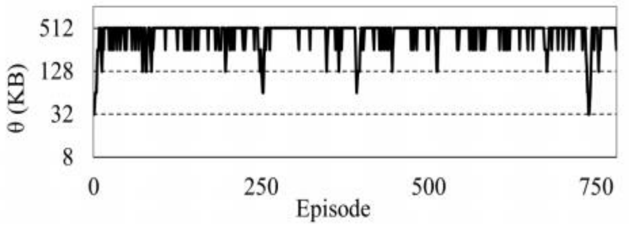


- RL outperforms all other techniques under most workloads
  - PC trace includes a larger number of hot data
  - In YCSB-A trace, most of the write requests are large and most of data are cold

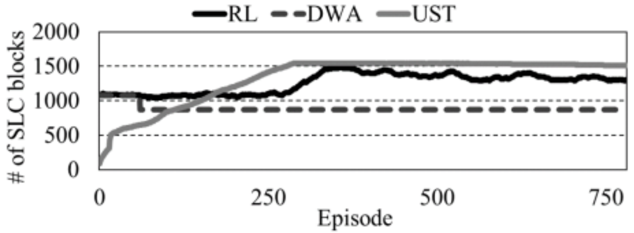
Change of SLC cache parameters



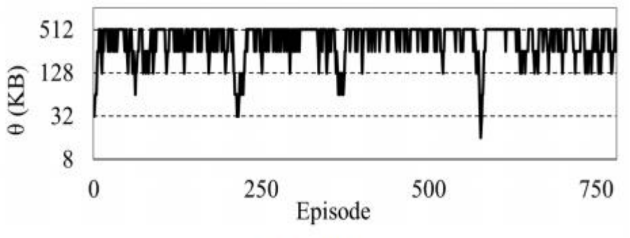
(a) PC



(a) PC



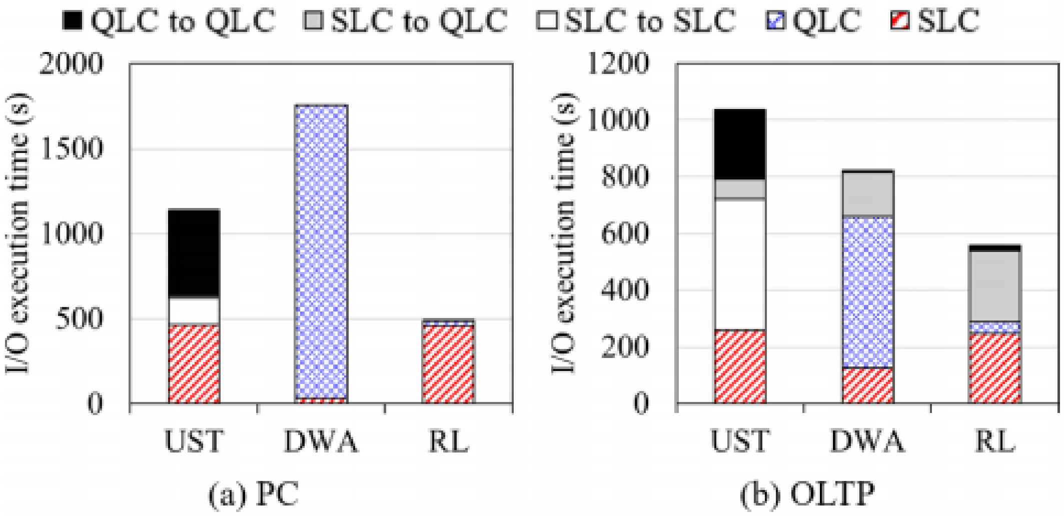
(b) OLTP



(b) OLTP

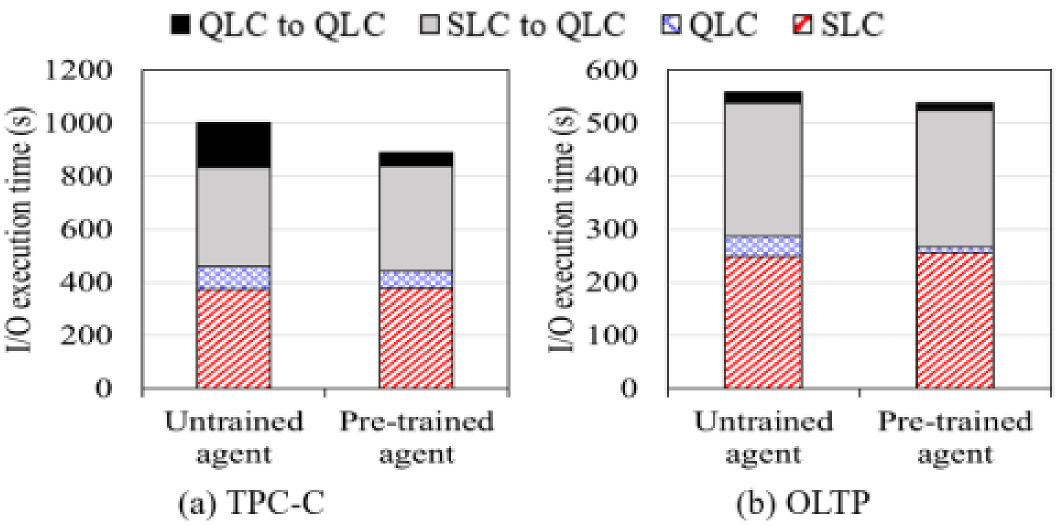
- The RL-based method adjusts more dynamically the SLC cache parameters
  - (PC trace) allocates a smaller number of SLC blocks than UST, but maintains a large value of θ

I/O Latency Breakdown



- 65.2% reduction at migration and garbage collection cost vs. UST
- Large QLC write overhead in DWA ➔ removed in the RL scheme

Effect of Agent Pre-training



- Pre-trained agent improves the write performance by up to 12.8% over untrained agent
  - can be applied quickly to a new system with a pre-trained agent

Conclusion

- Proposed an RL-based SLC cache technique
  - dynamically determines the optimal SLC cache parameters based on the system states
  - enhance write throughput and write amplification factor by 77.6% and 20.3% on average, respectively
  - without any prior knowledge about host workload or storage characteristics
- Future work
  - examine the effect of the proposed scheme at a real SSD
  - apply the technique at multi-stream SSDs