Challenges and Opportunities in Flash Storage Systems

Quality of Service on Flash

Ana Klimovic

PLATFORMLAB

Quality of service

Goal:

Enforce **tail read latency** & **throughput** guarantees on Flash

Quality of service

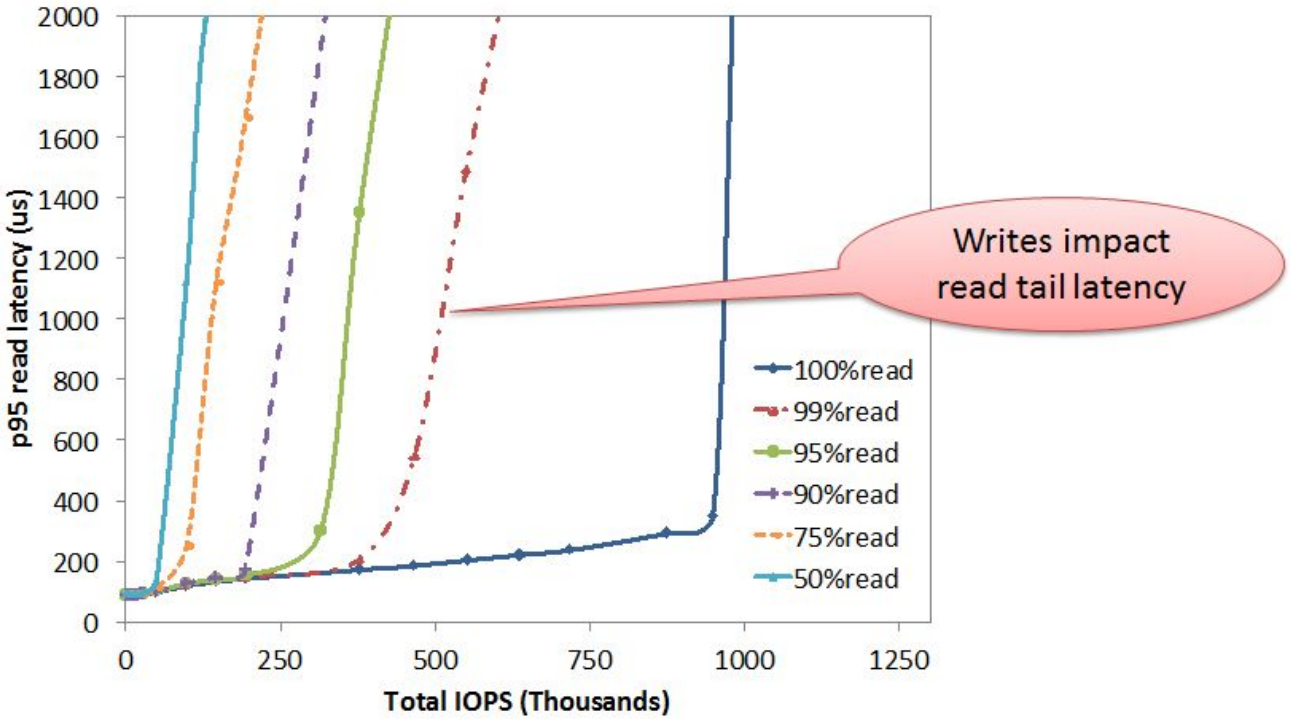
Goal:

Enforce **tail read latency** & **throughput** guarantees on Flash

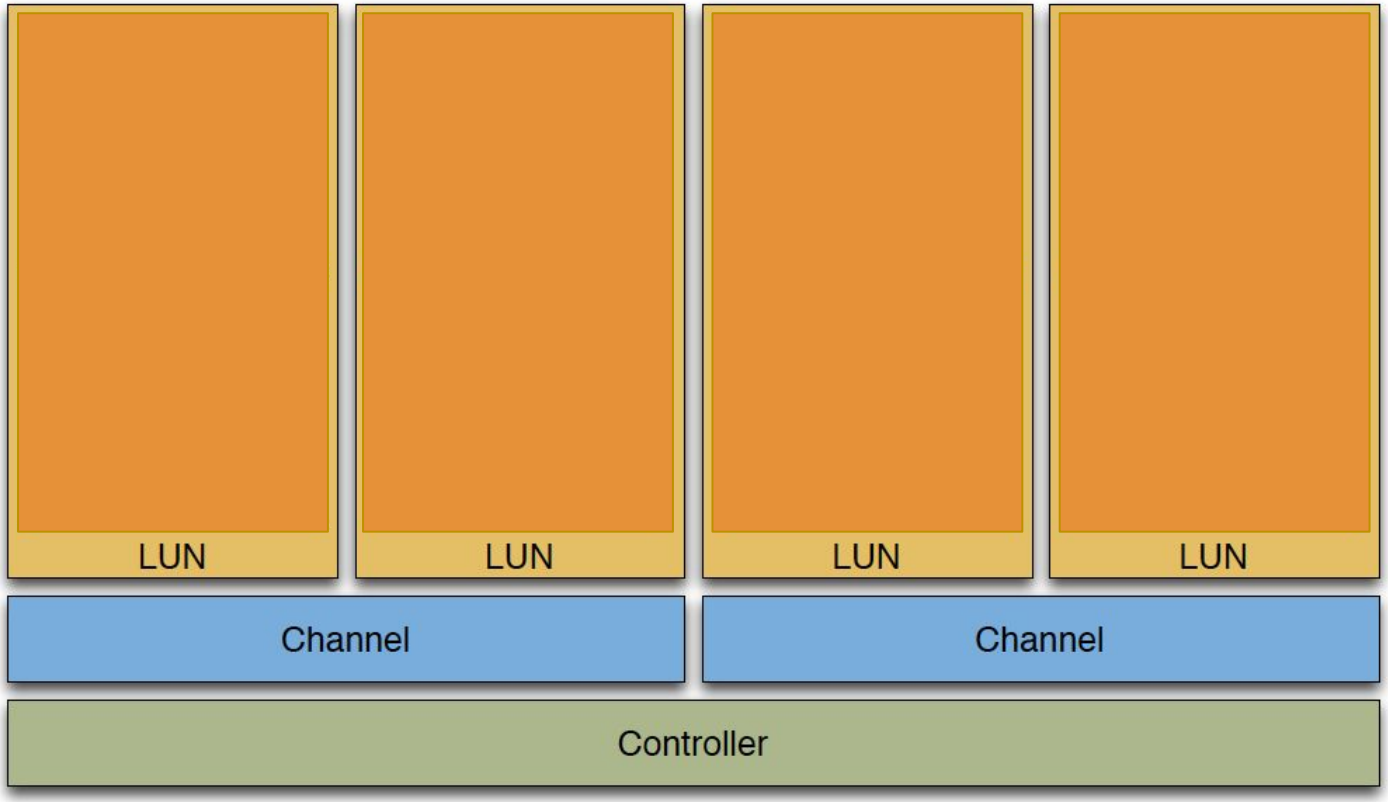
Challenges:

- Read/write interference
- Isolation between multiple tenants
- Flash device performance at high capacity utilization

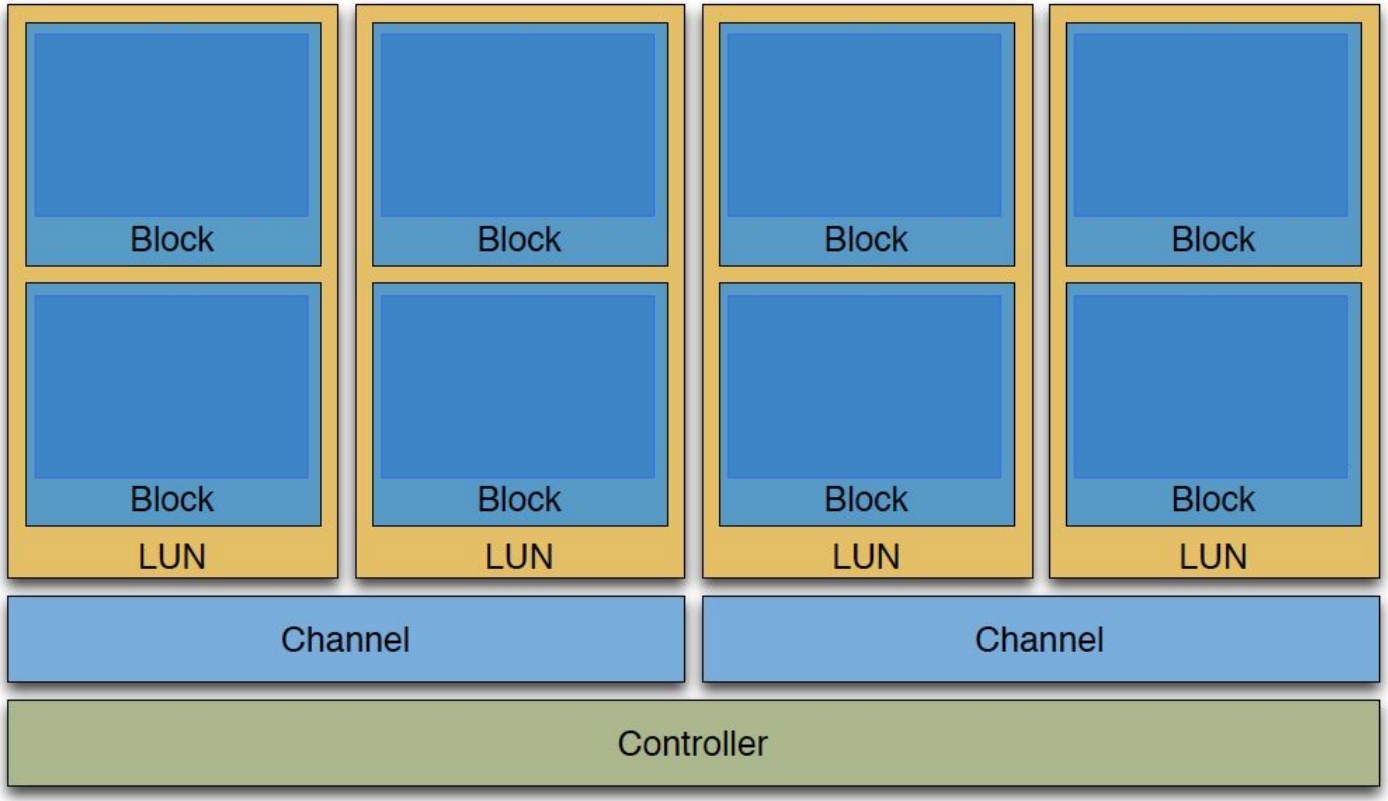
Read/write interference



Flash architecture

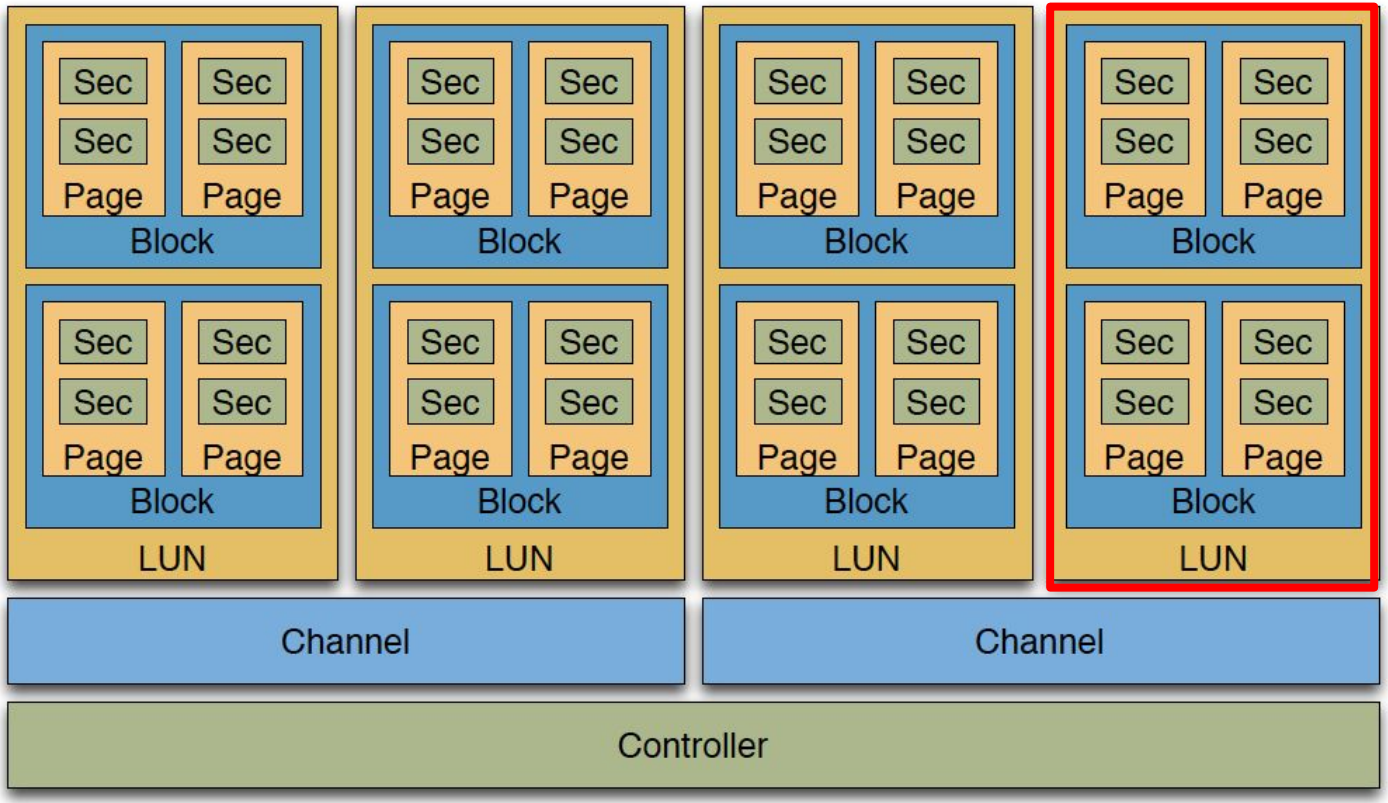


Flash architecture



Flash architecture

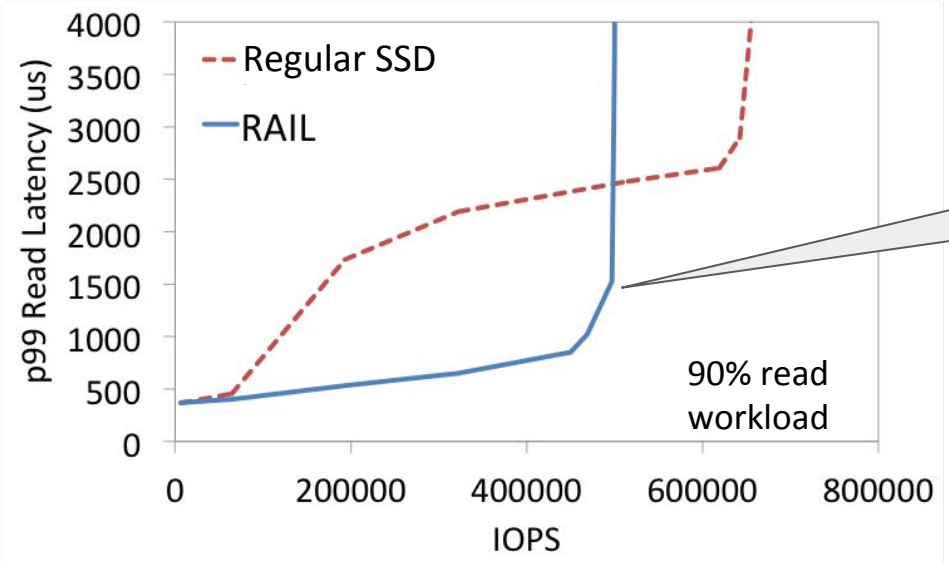
Each LUN can execute 1 operation at a time



	Latency
Read	70 us
Write	2 ms
Erase	5 ms

Idea: use **redundancy** to reduce read/write conflicts

- Store data pages with RAID-style parity across LUNs
- When a read conflicts with ongoing write, use alternative read path
  ⟶ reconstruct data based on parity



Tradeoff peak bandwidth for predictable latency

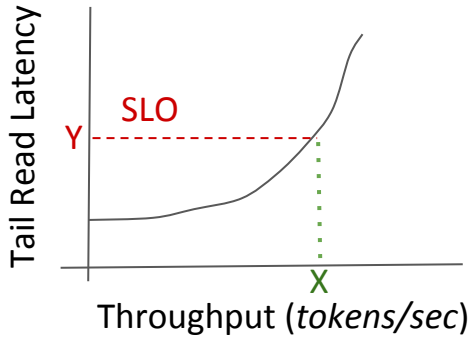
QoS on shared Flash

1. QoS-aware **physical data placement**

   - Isolate each tenant's data on separate physical blocks and/or LUNs
     e.g. Open-Channel SSD, Samsung multi-stream SSD

2. QoS-aware **I/O scheduling**

   - Model device performance & req cost
   - Token-based scheduling of
     latency-critical & best-effort traffic



Write Amplification

Assaf Eisenman

PLATFORMLAB

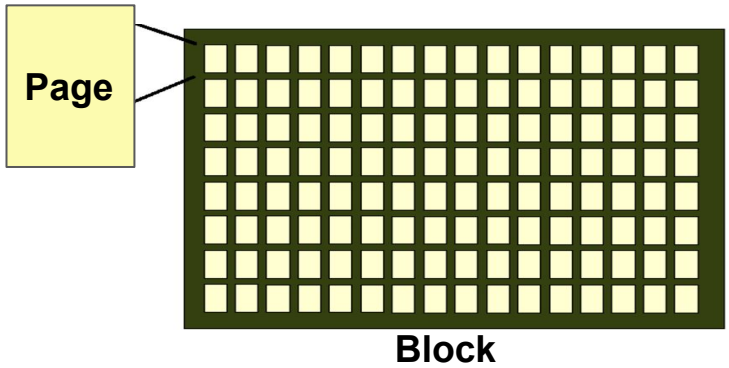
Write amplification

- The number of physical bits written to the SSD, for every bit written by the program

- Why write amplification is bad?
  - Durability: flash memory can be written and erased for a limited number of times
  - Performance: writes take longer

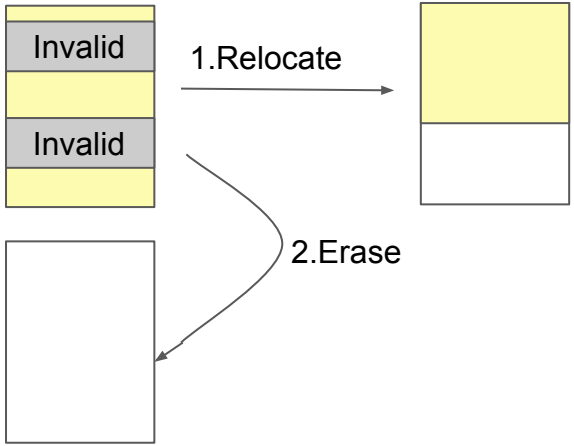
Why does flash suffer from write amplification?

- Read/Write operations are performed on **pages**
- Erase operations are performed on **blocks**
- Overwriting pages is not possible, they have to be erased first



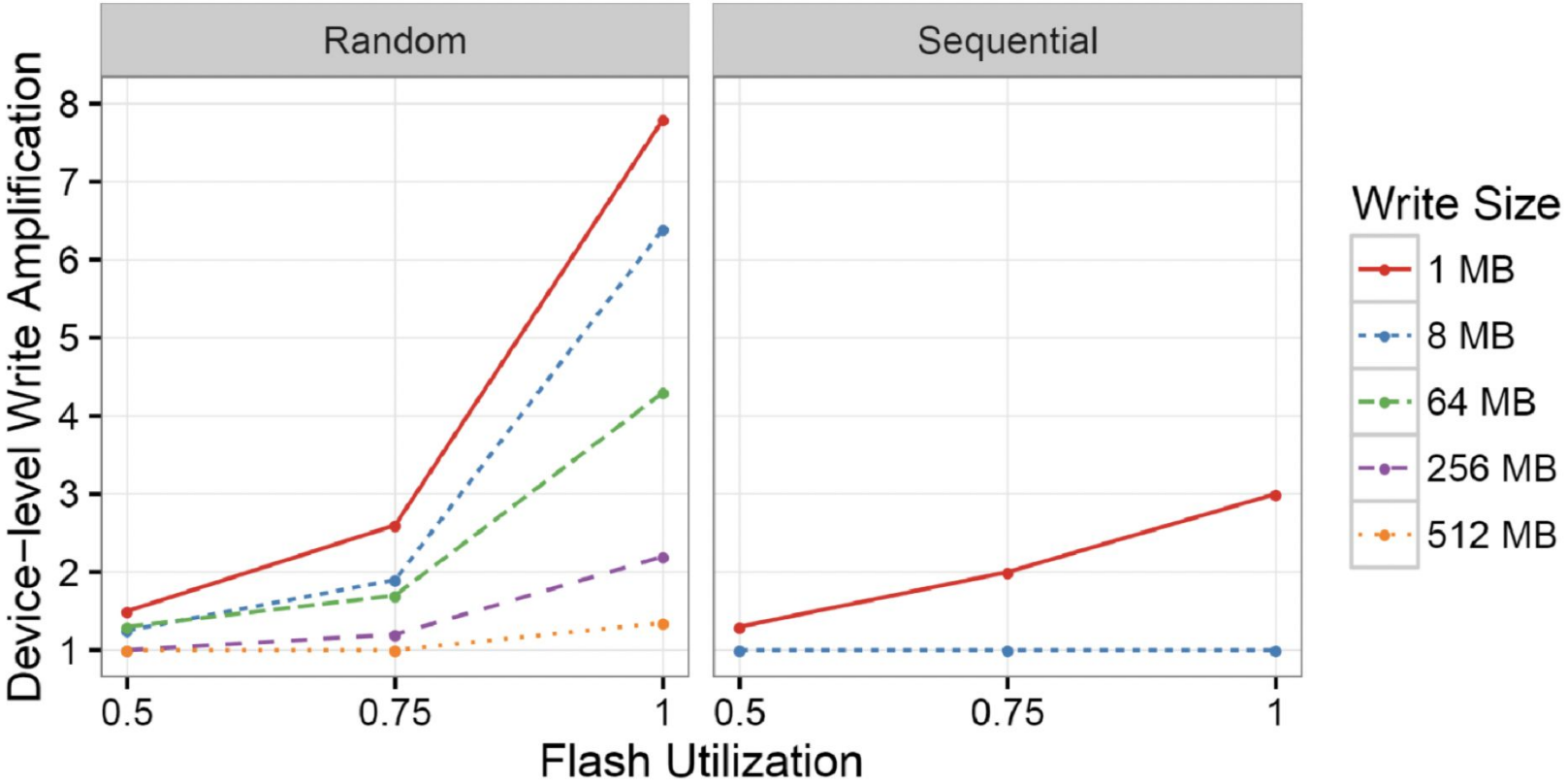
Why does flash suffer from write amplification?

- Garbage collection:



- Wear leveling
  - Trying to wear out all blocks over time as uniformly as possible

An experiment on write amplification



How can we improve write amplification?

- Over-provision (low utilization)
- Write sequentially
- Write in big chunks

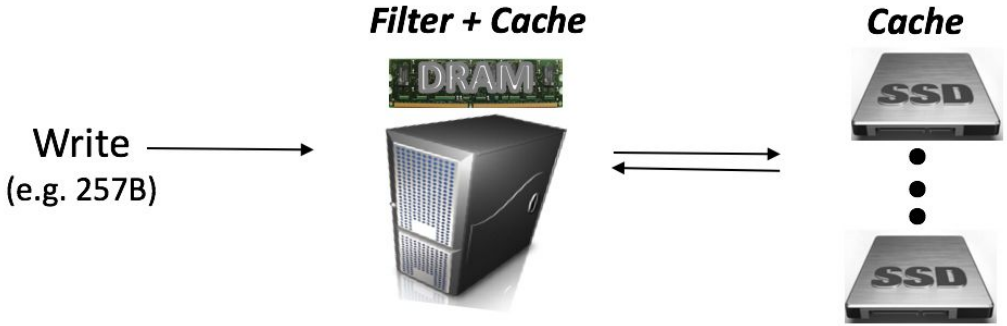
➔ Not feasible for many applications

Flashield

- Widely-used key-value caches (Redis, Memcached) are based on DRAM
- **Challenge**: Cache workload quickly degrades SSD durability
- **Solution**: Flashield is first general-purpose SSD key-value cache with same lifetime as DRAM

Flashield

- Key insight: not all objects are flash-worthy (“flashy”)
  - Use DRAM as a filter for “flashiness”
- Ideal candidates for flash:
  - Immutable in the near future
  - Frequently read in the future
- Leverage machine learning to reduce write amplification by 5-25x



Flashield Results