Final Optimization

Team 9

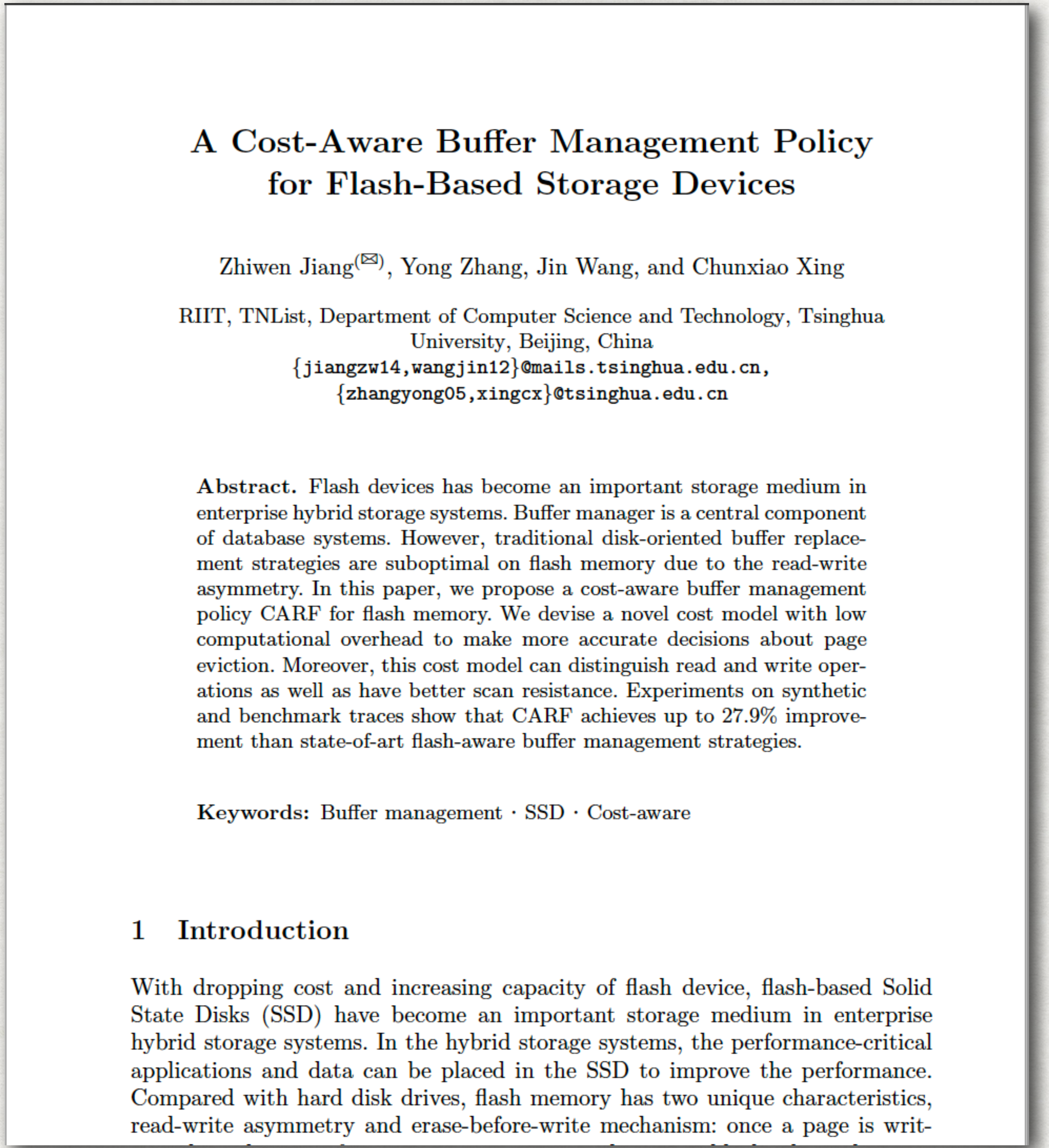
106062307 林奕鑫

106062322 江岷錡

Implementation

Optimize Throughput

- Flashed-Based Storage Devices on Buffer Replacement (Publication)
- Shared Calvin Transaction Cache
- …



A Cost-Aware Buffer Management Policy for Flash-Based Storage Devices

Zhiwen Jiang(✉), Yong Zhang, Jin Wang, and Chunxiao Xing

RIIT, TNList, Department of Computer Science and Technology, Tsinghua University, Beijing, China
{jiangzw14,wangjin12}@mails.tsinghua.edu.cn,
{zhangyong05,xingcx}@tsinghua.edu.cn

Abstract. Flash devices has become an important storage medium in enterprise hybrid storage systems. Buffer manager is a central component of database systems. However, traditional disk-oriented buffer replacement strategies are suboptimal on flash memory due to the read-write asymmetry. In this paper, we propose a cost-aware buffer management policy CARF for flash memory. We devise a novel cost model with low computational overhead to make more accurate decisions about page eviction. Moreover, this cost model can distinguish read and write operations as well as have better scan resistance. Experiments on synthetic and benchmark traces show that CARF achieves up to 27.9% improvement than state-of-art flash-aware buffer management strategies.

Keywords: Buffer management · SSD · Cost-aware

1 Introduction

With dropping cost and increasing capacity of flash device, flash-based Solid State Disks (SSD) have become an important storage medium in enterprise hybrid storage systems. In the hybrid storage systems, the performance-critical applications and data can be placed in the SSD to improve the performance. Compared with hard disk drives, flash memory has two unique characteristics, read-write asymmetry and erase-before-write mechanism: once a page is writ-

Characteristic of SSD

Read-write asymmetry

erase-before-write mechanism

~~Temporal locality~~

~~Minimize miss rate~~

Total I/O cost

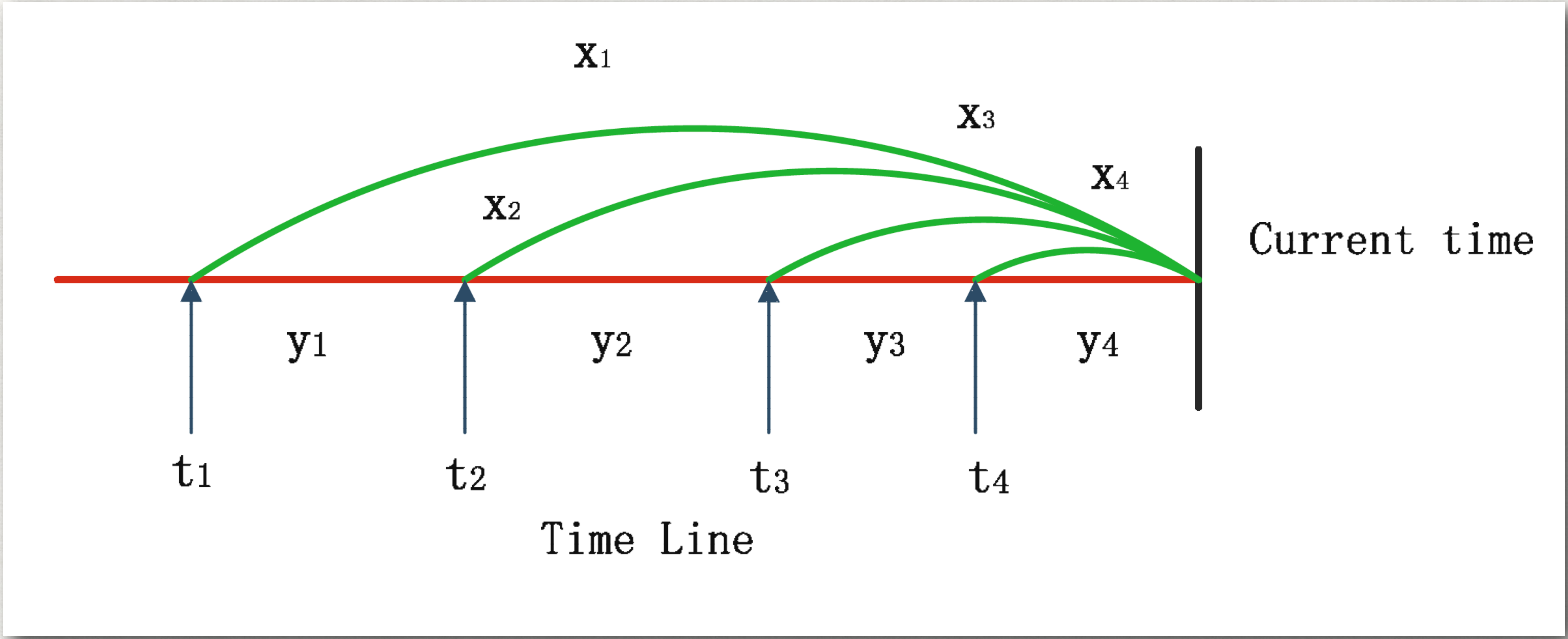
CARF

Cost-aware Algorithm combining Recency and Frequency

Recency and Frequency

Weight ➢ Eviction

Low Overhead



x_1
x_3
x_4
x_2
Current time
y_1
y_2
y_3
y_4
t_1
t_2
t_3
t_4
Time Line

$$R_{\mathrm{t_c}}(p) = \begin{cases} R, \textit{lastdirty}(p) = \textit{false} \wedge \text{ modified at } t_{\mathrm{c}} \\ 1, \ \text{otherwise} \end{cases}$$

$$W(x) = a^x, (0 < a \leq 1) \ .$$

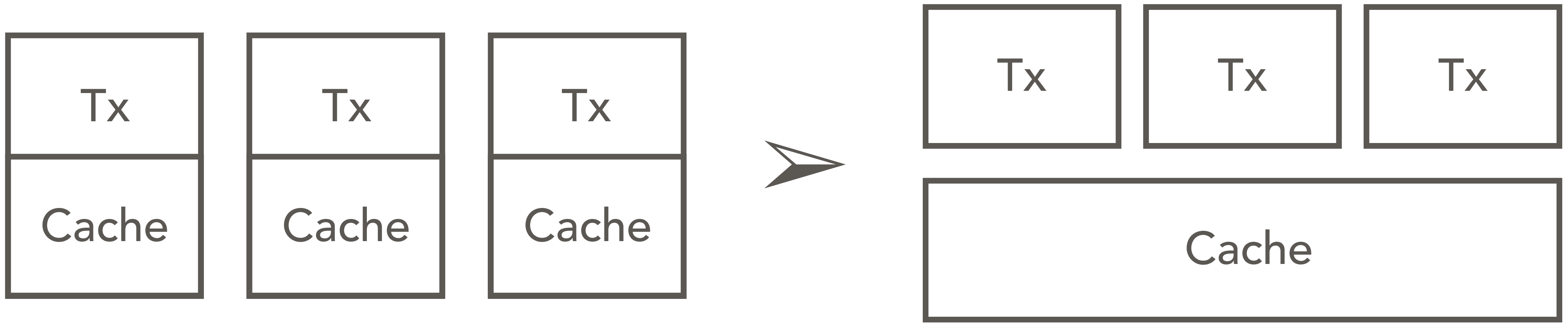
$$FPW_{\mathrm{t_c}}(p) = R_{\mathrm{t_c}}(p) * [W(0) + W(\mathrm{t_c} - \textit{Lasttime}(p)) * \textit{Lastweight}(p)] \ .$$

Difficulty

Buffer Victim	Weight Memory ➢ Min Heap (O(1)、O(logn))
	Select Victim & Buffer Replacement (Index、Recovery….Immediately unpin)
	Find Optimal Algorithm Parameter
Integration	Modify All Buffer Connection & Interface
	Make BufferMgr Aware of Buffer State

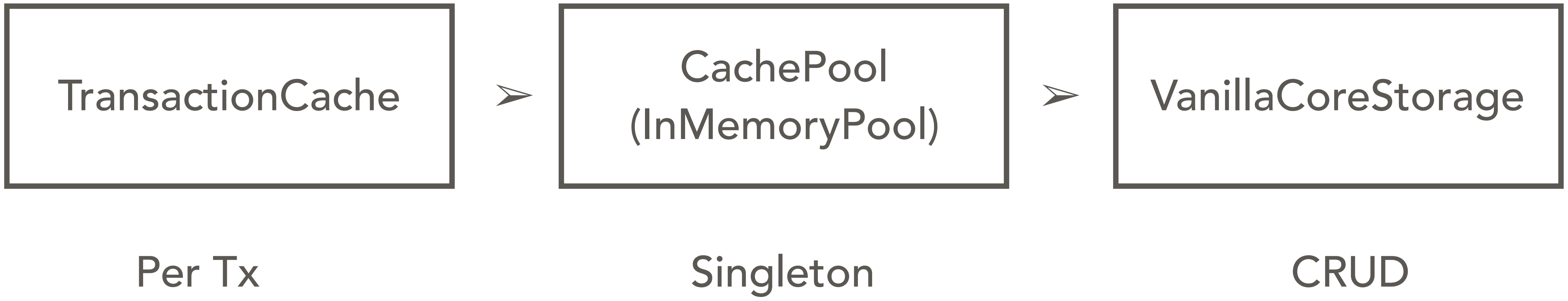
Shared Calvin Cache

Transactions share same cache



Difficulty

Architecture



Difficulty

Algorithm

Pin-time: GetRecord / SetRecord
Unpin-time: flush

Lock Awareness

Cache Replacement (Clock-wise)

Flush

Tx finish

No enough space

Waiting (5000)

Flush Self-Data (notify all)

Re-Fetch Space