

# NVRAM-aware Logging in Transaction Systems

---

**Jian Huang**

Karsten Schwan

Moinuddin K. Qureshi



# Logging Support for Transactions

---



The image displays the acronym ACID, where each letter is a different color and has its corresponding property written below it in a smaller, italicized font. The 'A' is red and stands for Atomicity, the 'C' is green for Consistency, the 'I' is blue for Isolation, and the 'D' is maroon for Durability. The entire graphic is enclosed in a rounded rectangular border.

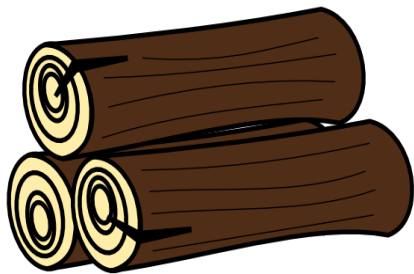
**A** **C** **I** **D**  
*ATOMICITY* *CONSISTENCY* *ISOLATION* *DURABILITY*

# Logging Support for Transactions

---

**A** **C** **I** **D**  
*ATOMICITY* *CONSISTENCY* *ISOLATION* *DURABILITY*

## ARIES: Disk-Based Approach (TODS'92)

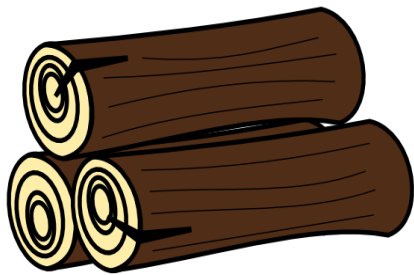


Write-ahead Logging

# Logging Support for Transactions

**A** **C** **I** **D**  
*ATOMICITY* *CONSISTENCY* *ISOLATION* *DURABILITY*

## ARIES: Disk-Based Approach (TODS'92)



Write-ahead Logging

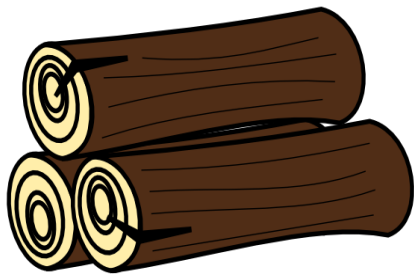


Rollback

# Logging Support for Transactions

**A** **C** **I** **D**  
*ATOMICITY* *CONSISTENCY* *ISOLATION* *DURABILITY*

## ARIES: Disk-Based Approach (TODS'92)



Write-ahead Logging

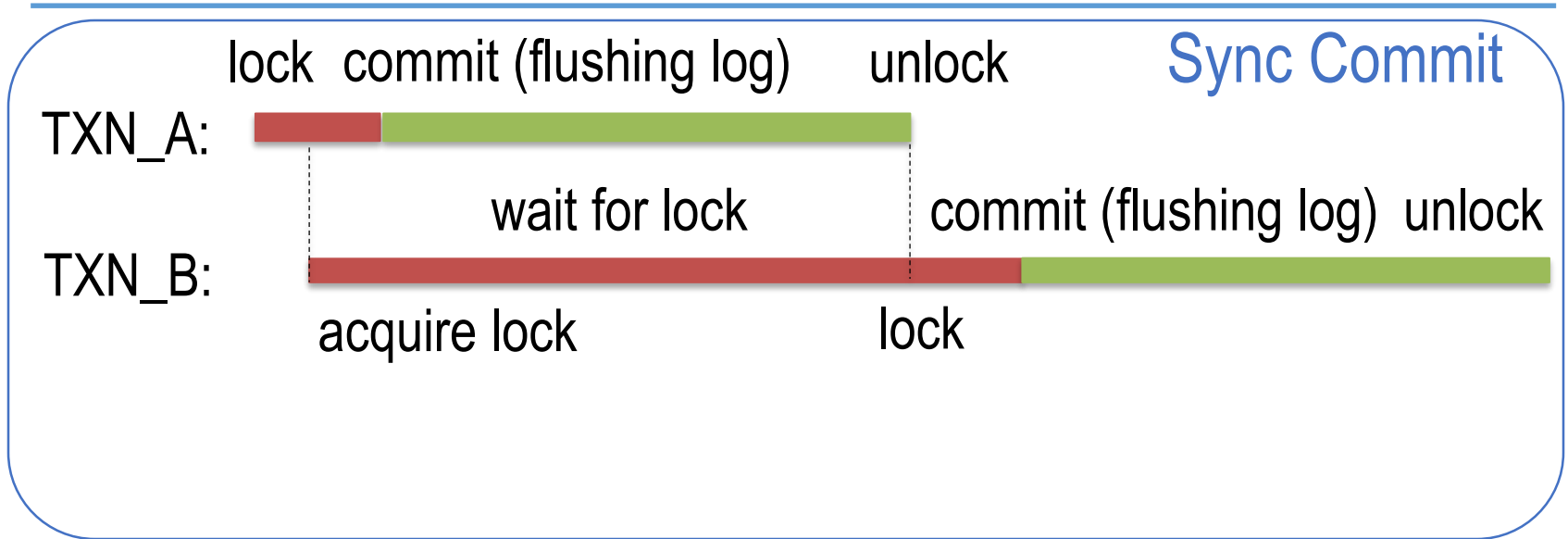


Rollback

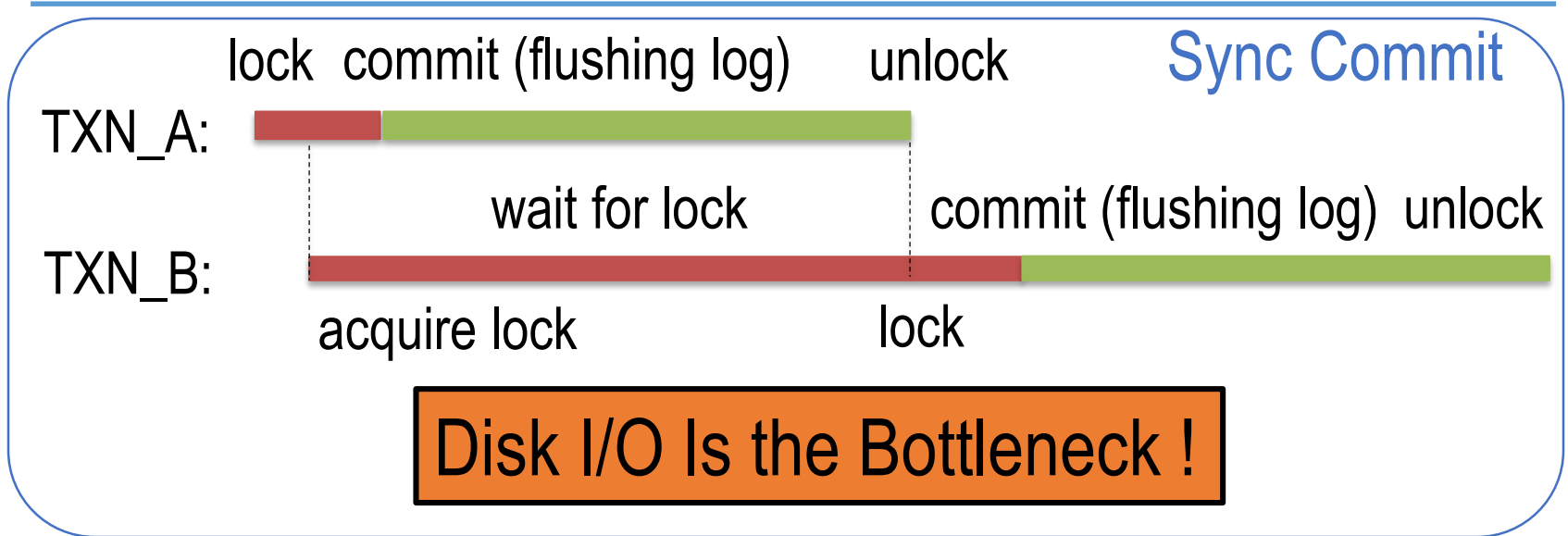


Centralized Log Buffer

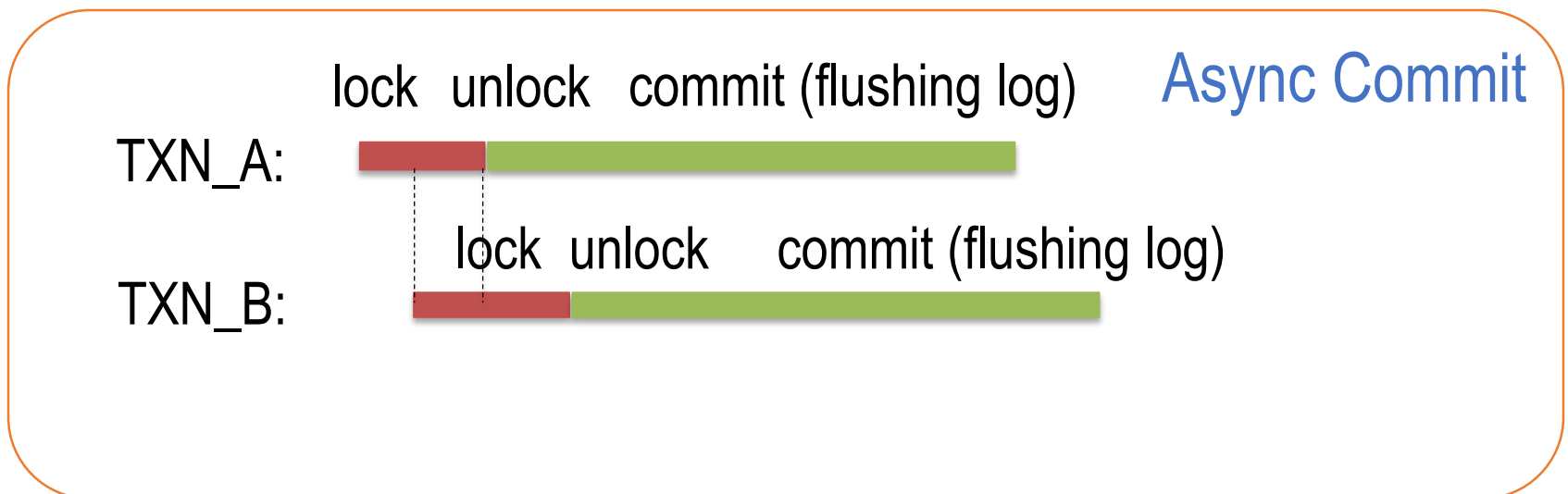
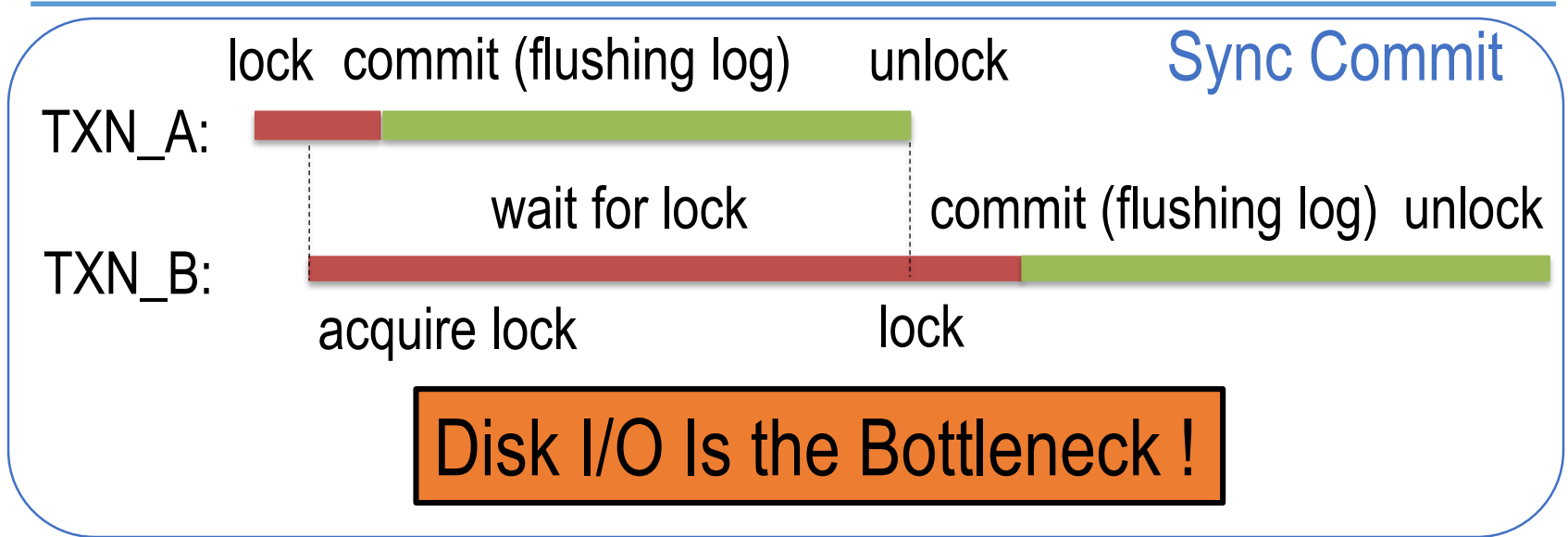
# On the Logging Persistence



# On the Logging Persistence

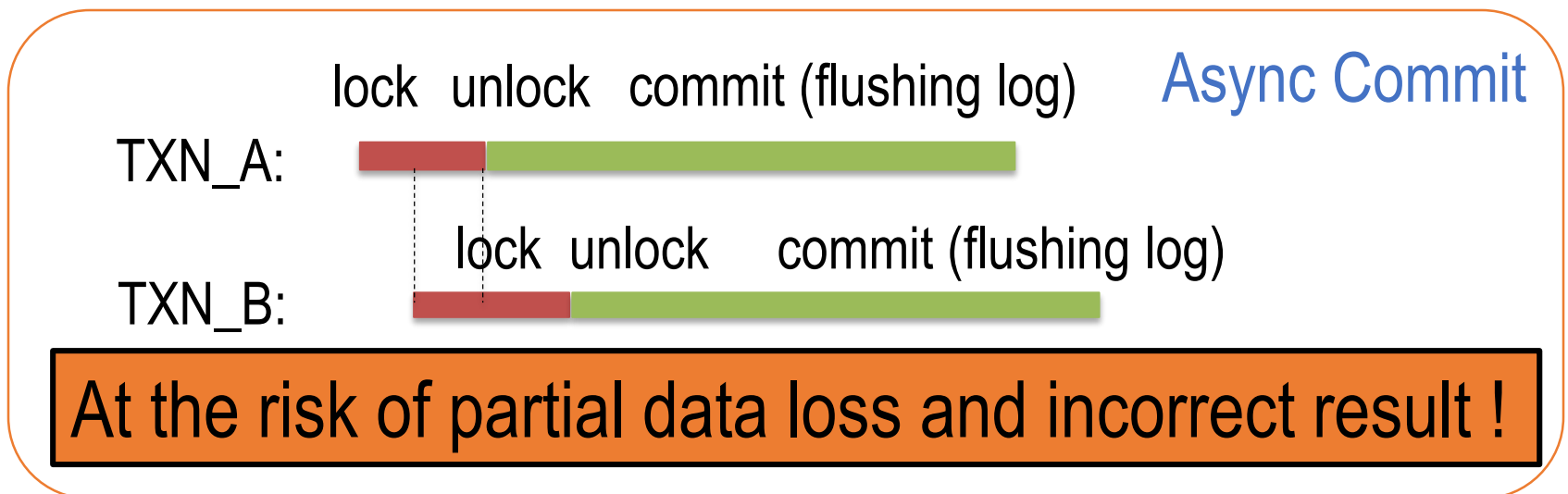
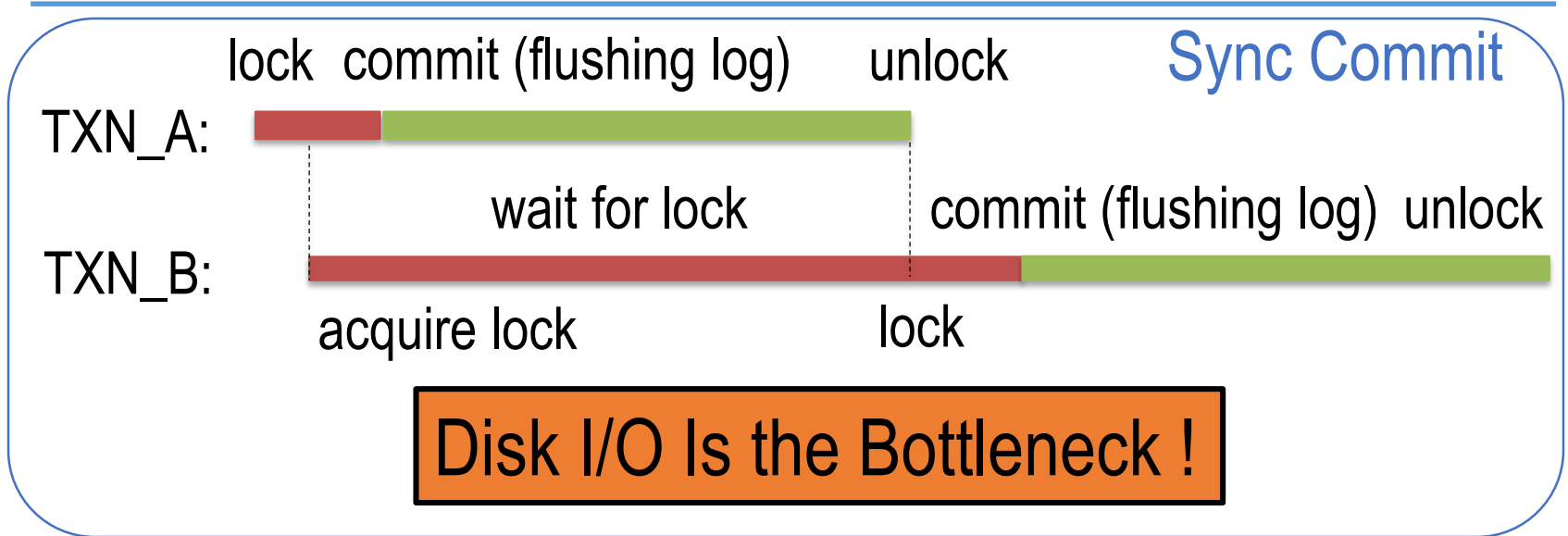


# On the Logging Persistence





# On the Logging Persistence



# Rethink the Logging with NVRAM

---



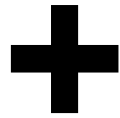
Close to DRAM

# Rethink the Logging with NVRAM

---



Close to DRAM



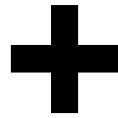
Byte Addressability

# Rethink the Logging with NVRAM

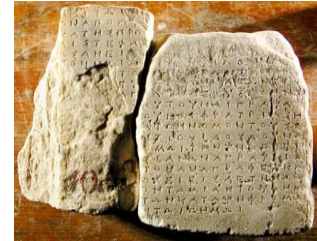
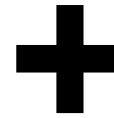
---



Close to DRAM



Byte Addressability

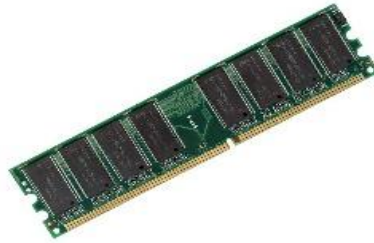
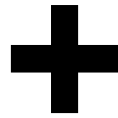


Non-Volatility

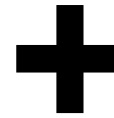
# Rethink the Logging with NVRAM



Close to DRAM



Byte Addressability



Non-Volatility



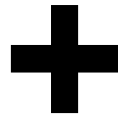
## Disk-based logging

- Avoid frequent disk access
- Obtain sequential access pattern

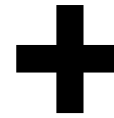
# Rethink the Logging with NVRAM



Close to DRAM



Byte Addressability



Non-Volatility



## Disk-based logging

- Avoid frequent disk access
- Obtain sequential access pattern



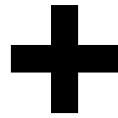
## Flash-based logging

- Reduce log flushing time
- Atomic Write (OSDI'08)

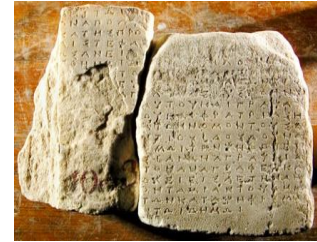
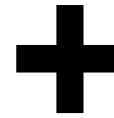
# Rethink the Logging with NVRAM



Close to DRAM



Byte Addressability



Non-Volatility



Disk-based logging

- Avoid frequent disk access
- Obtain sequential access pattern



Flash-based logging

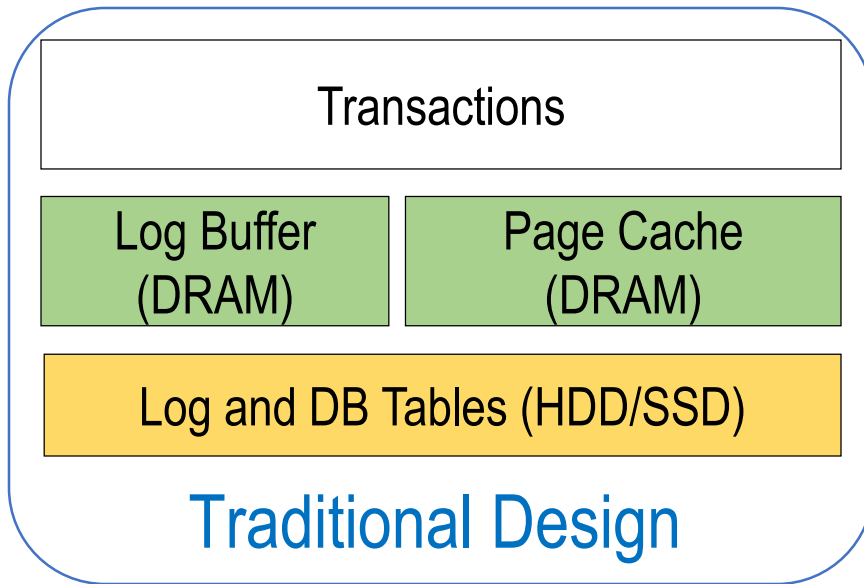
- Reduce log flushing time
- Atomic Write (OSDI'08)



NVRAM-based

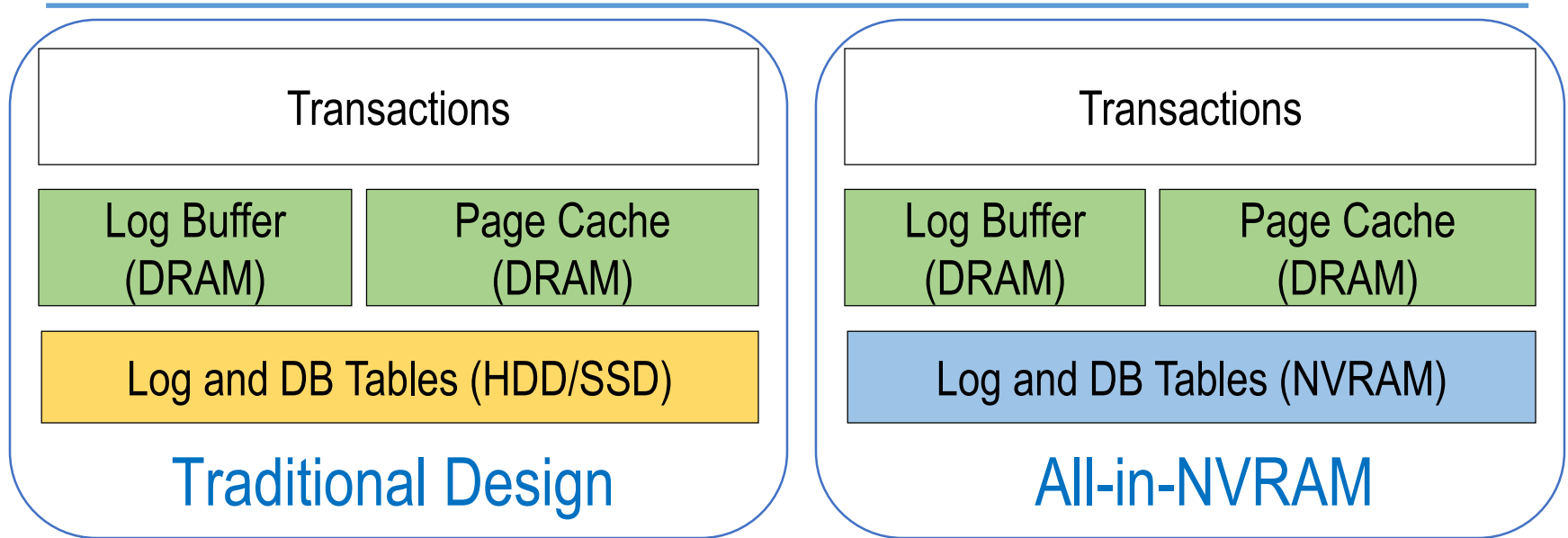
# How to Use NVRAM in DB System?

---

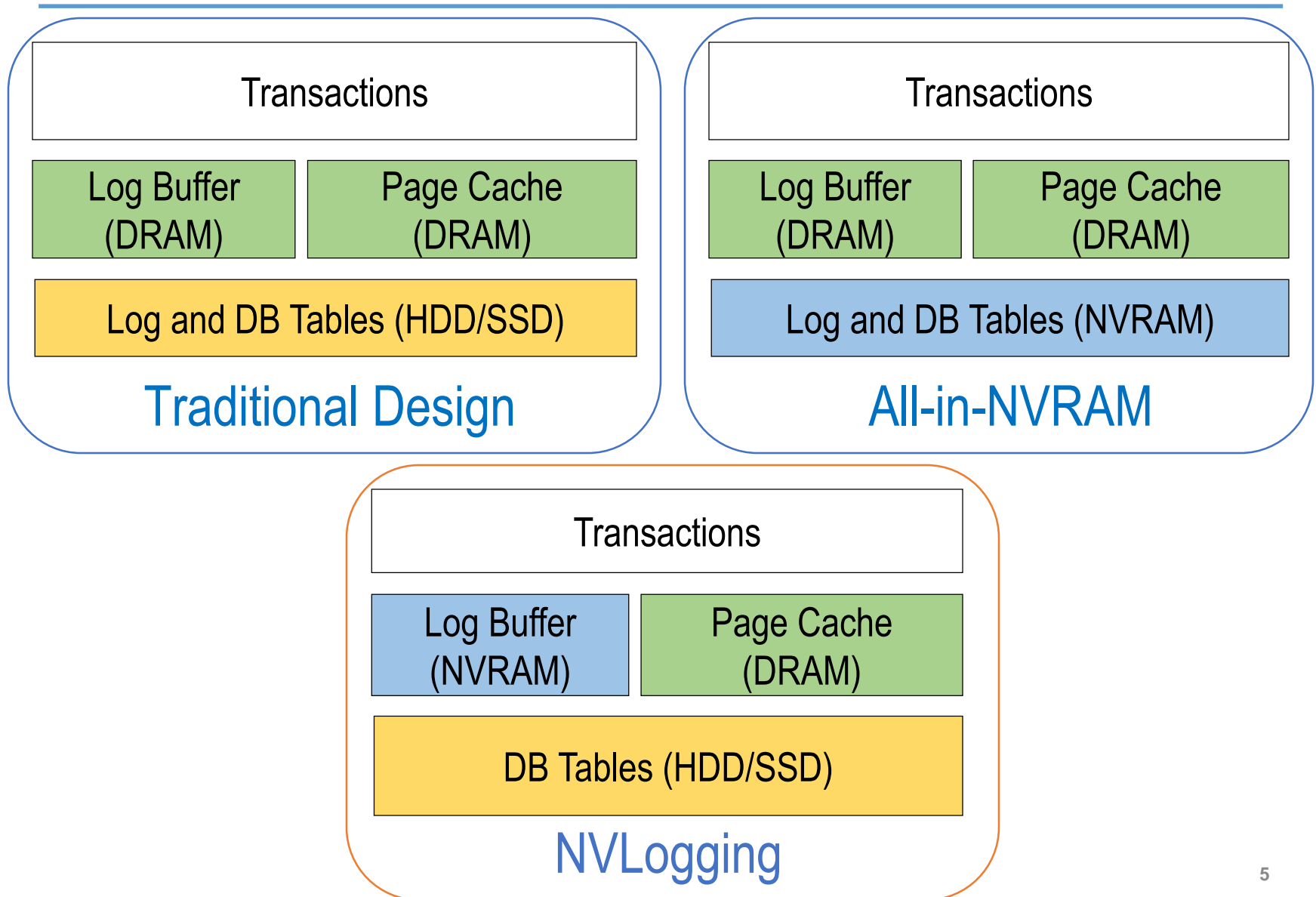




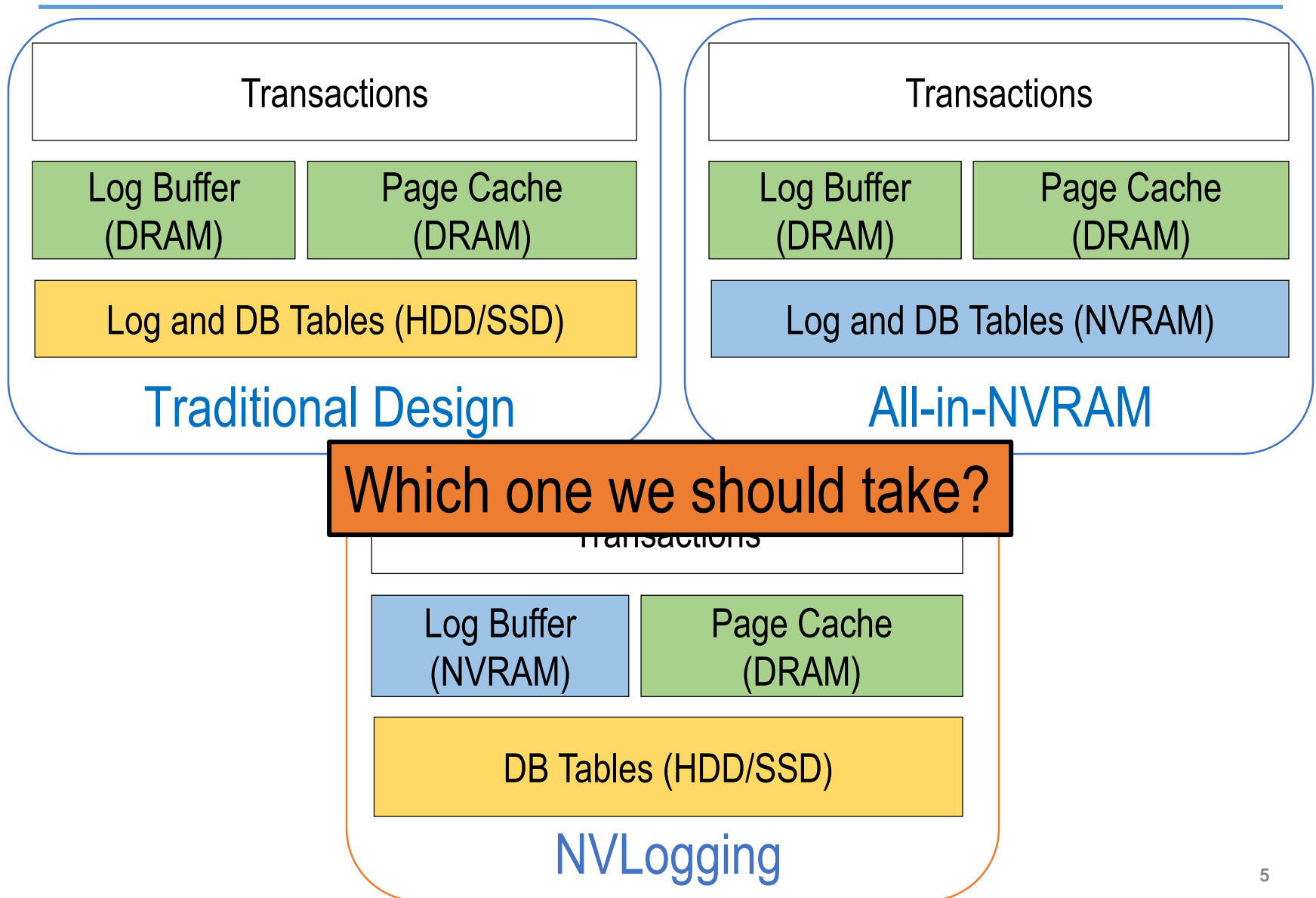
# How to Use NVRAM in DB System?



# How to Use NVRAM in DB System?



# How to Use NVRAM in DB System?



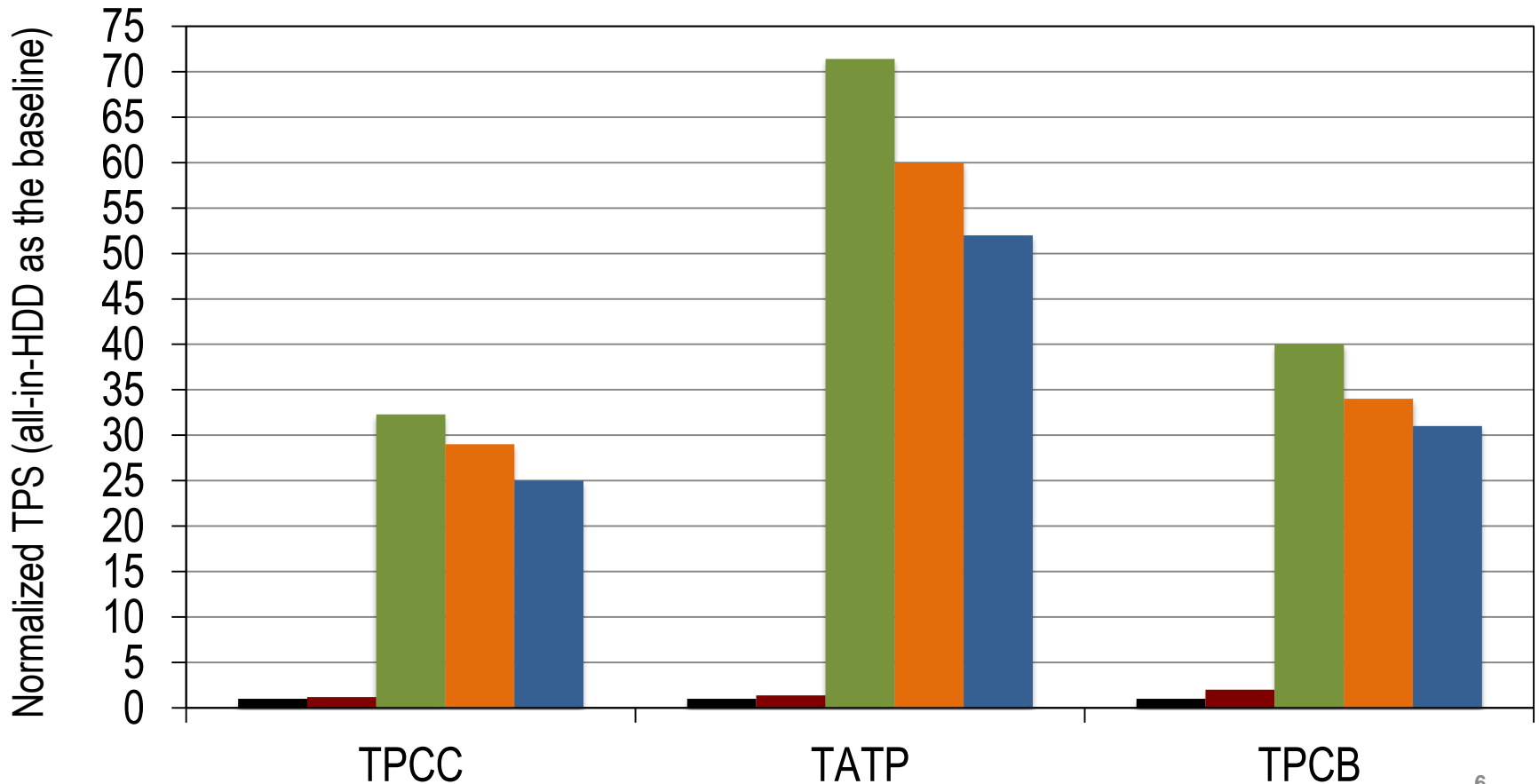
# Using NVRAM in Cost-Effective Way

---

# Using NVRAM in Cost-Effective Way

■ all-in-HDD

■ all-in-SSD

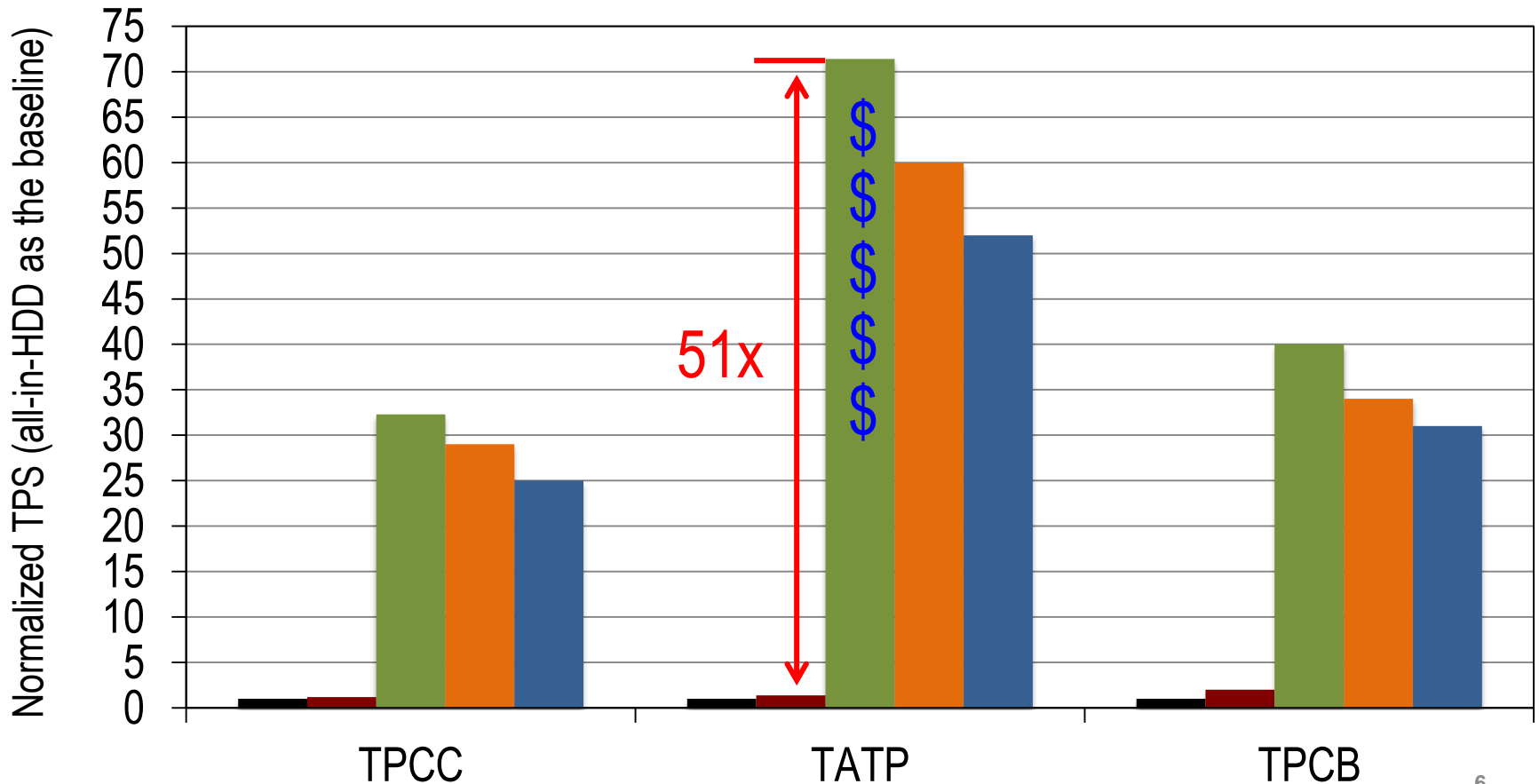


# Using NVRAM in Cost-Effective Way

■ all-in-HDD

■ all-in-SSD

■ all-in-NVRAM



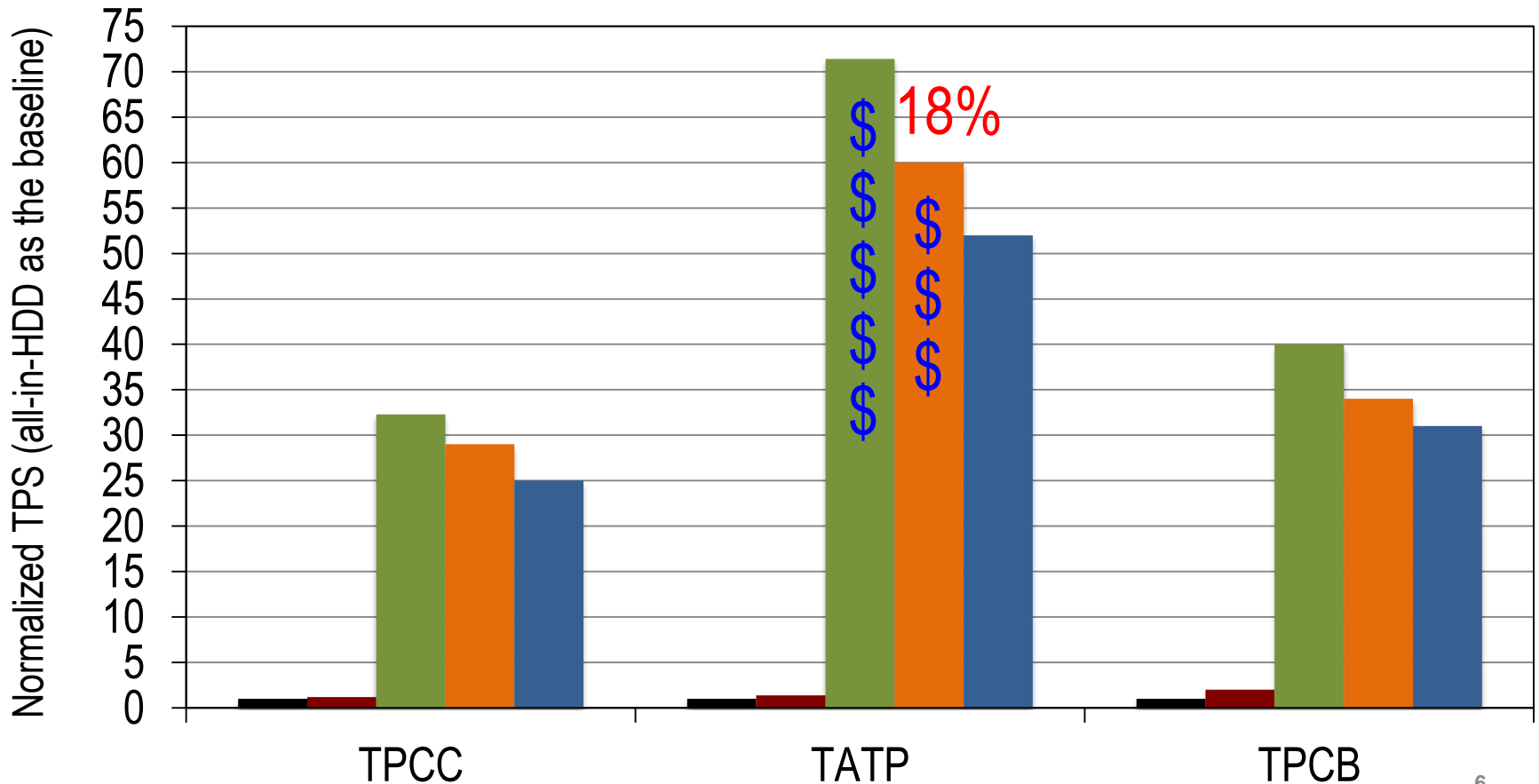
# Using NVRAM in Cost-Effective Way

■ all-in-HDD

■ all-in-NVRAM

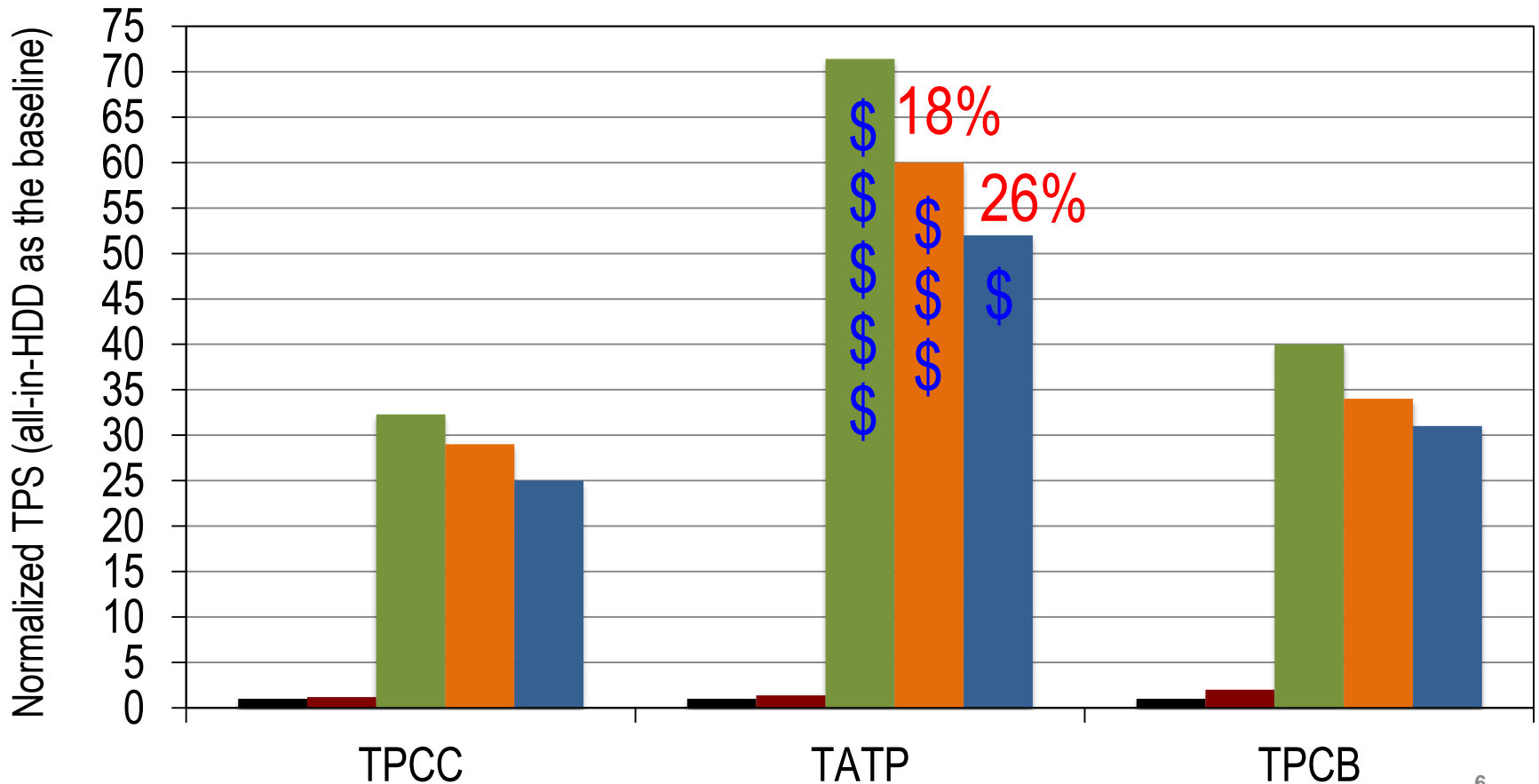
■ all-in-SSD

■ db-in-SSD, log-in-NVRAM



# Using NVRAM in Cost-Effective Way

- all-in-HDD
- all-in-SSD
- all-in-NVRAM
- db-in-SSD, log-in-NVRAM
- db-in-HDD, log-in-NVRAM





# Using NVRAM in Cost-Effective Way

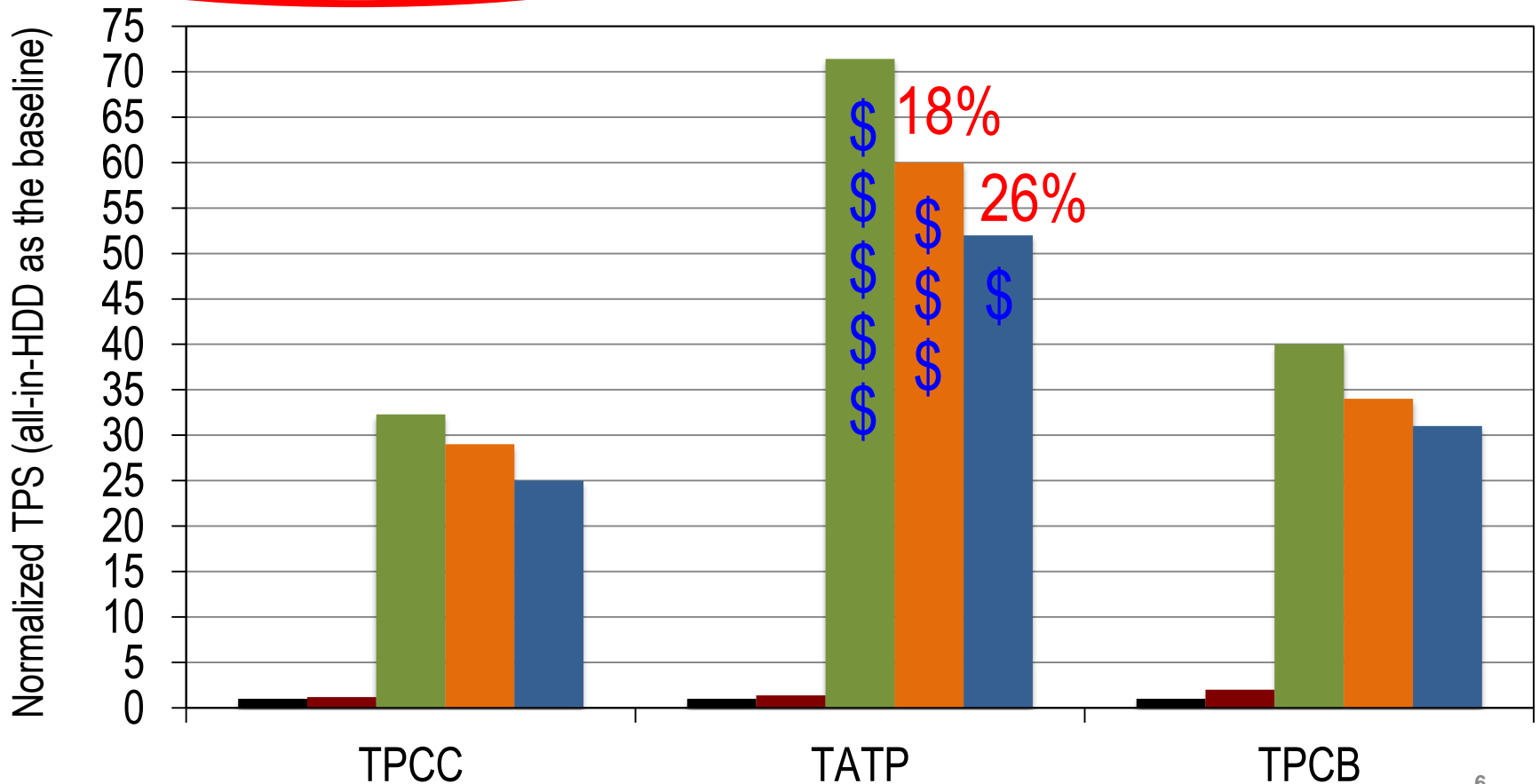
■ all-in-HDD

■ all-in-NVRAM

■ db-in-HDD, log-in-NVRAM

■ all-in-SSD

■ db-in-SSD, log-in-NVRAM



# Using NVRAM in Cost-Effective Way

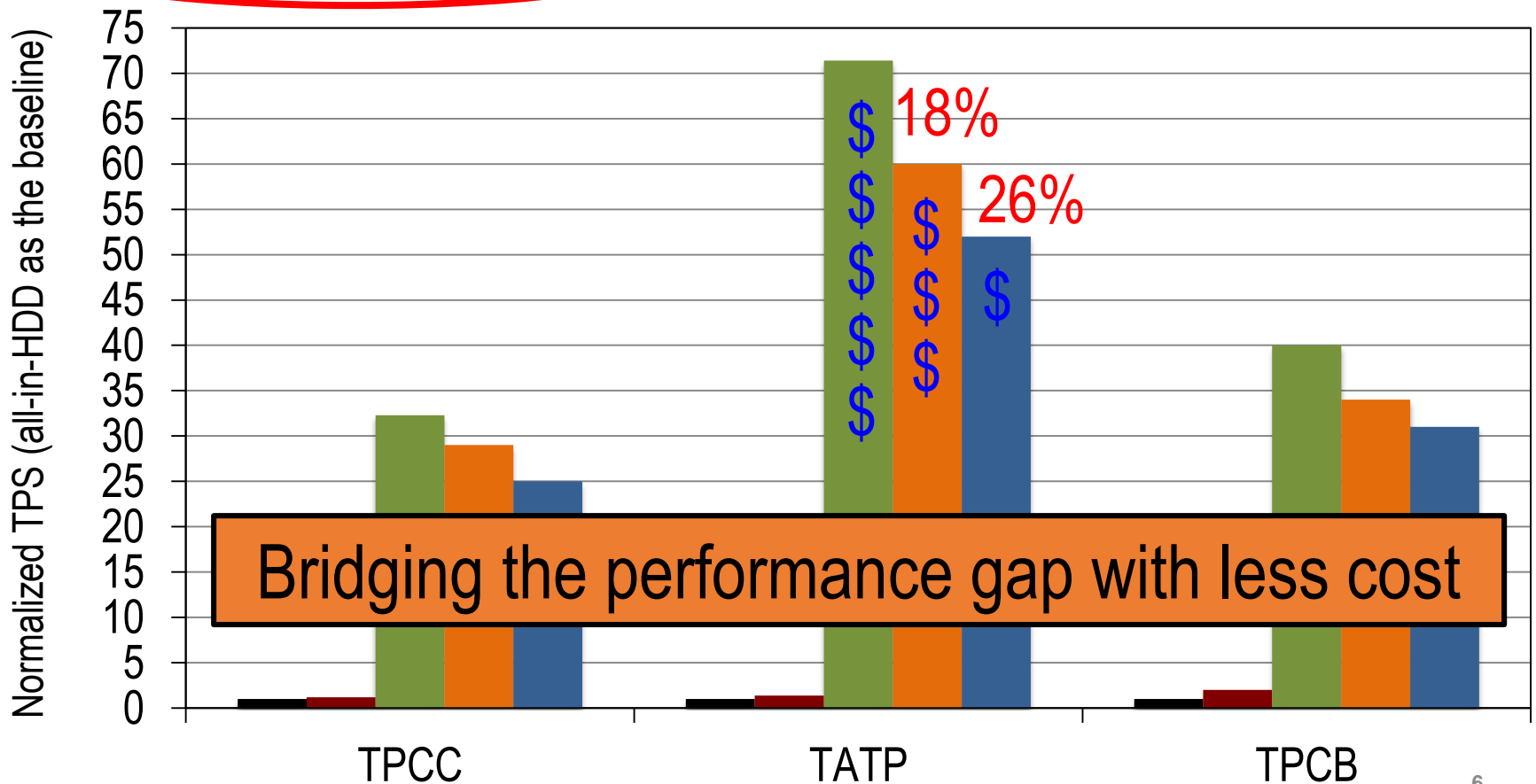
■ all-in-HDD

■ all-in-NVRAM

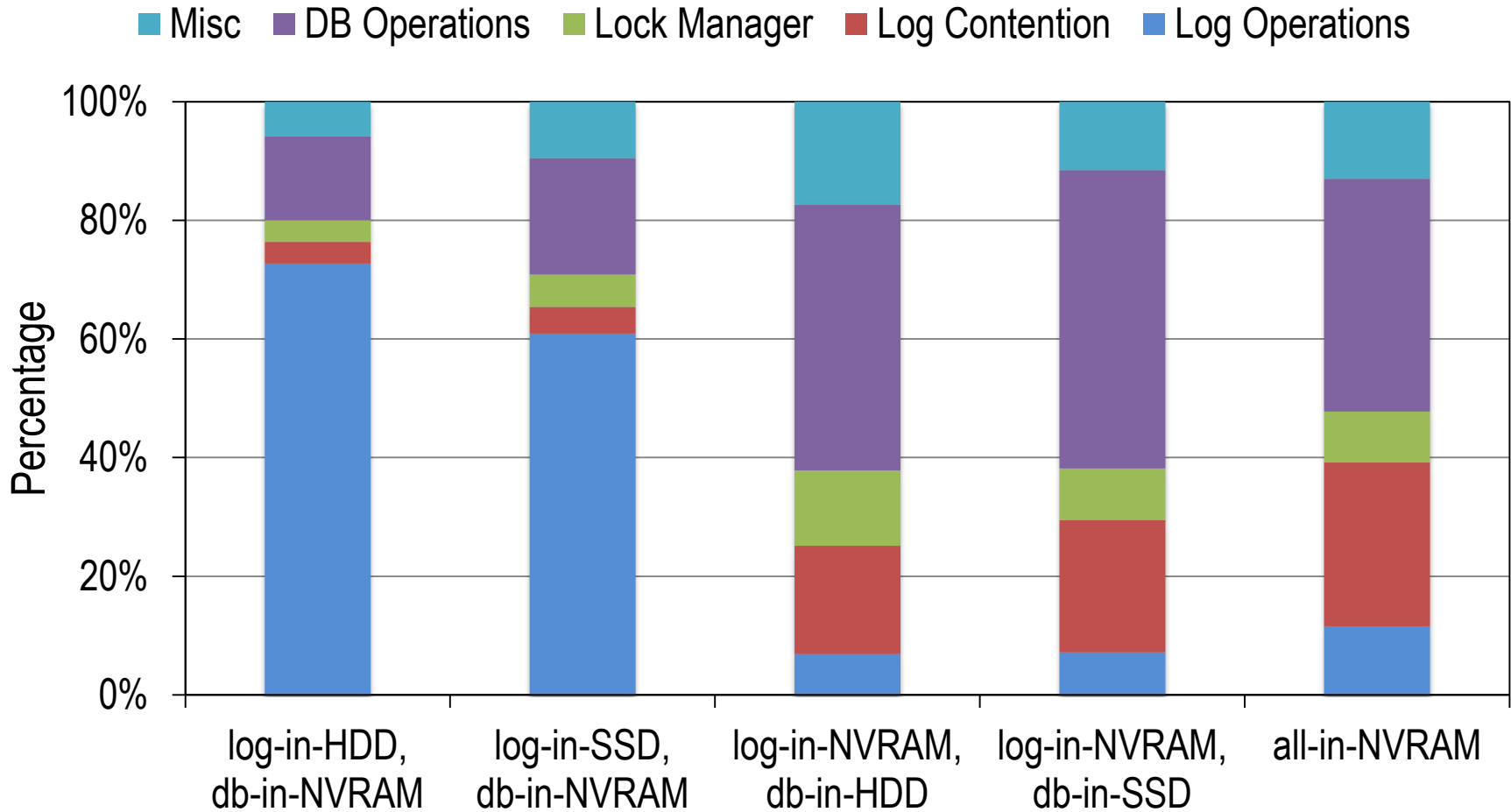
■ db-in-HDD, log-in-NVRAM

■ all-in-SSD

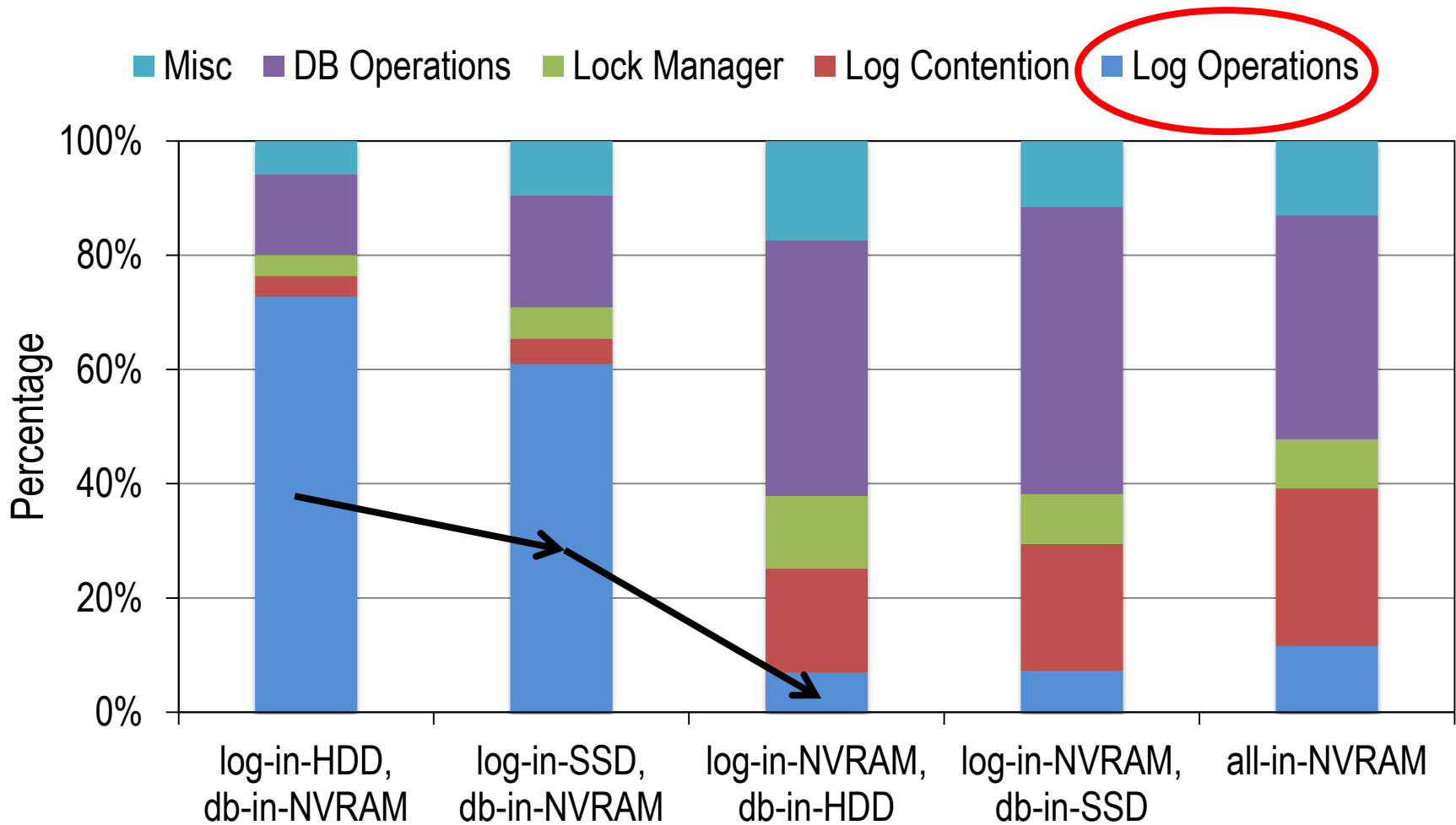
■ db-in-SSD, log-in-NVRAM



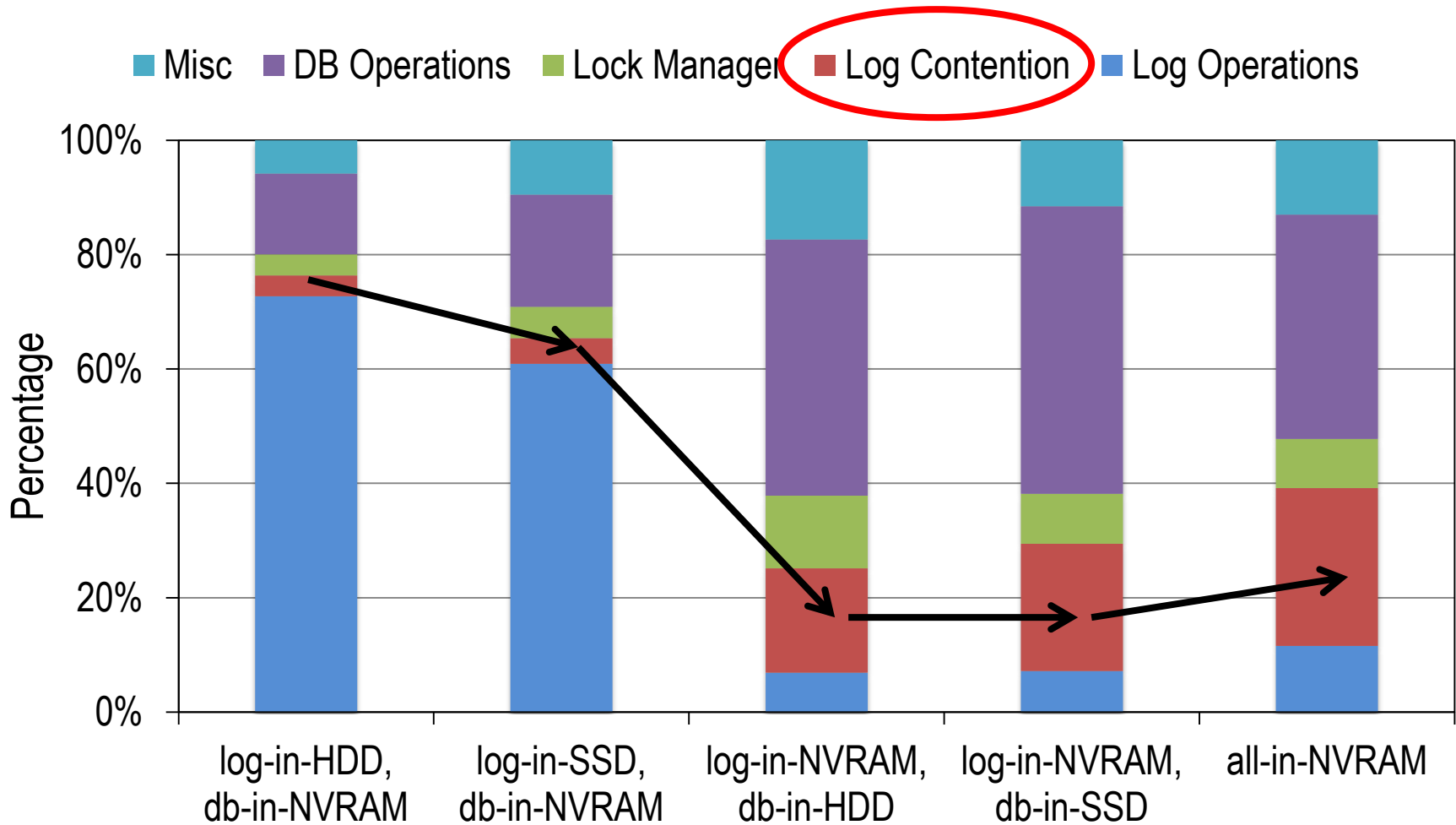
# Bottlenecks Shifted from I/O to Software



# Bottlenecks Shifted from I/O to Software



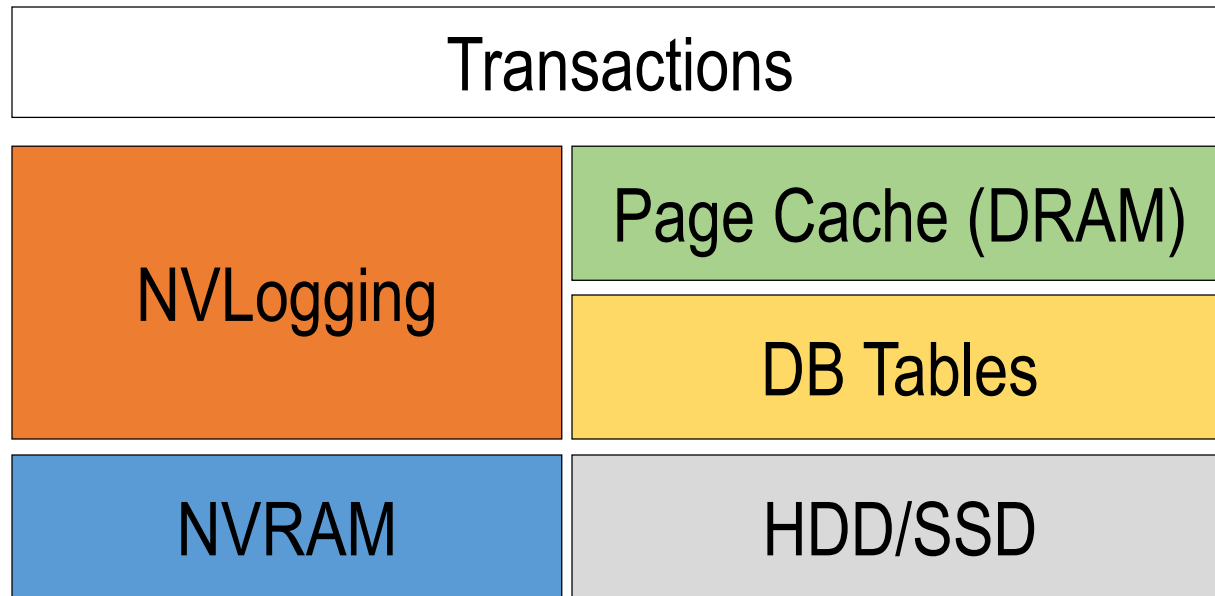
# Bottlenecks Shifted from I/O to Software



Software-induced overhead becomes the new bottleneck !

# NVLogging: Redesign the Logging with NVRAM

---



- 1 Use NVRAM in Cost-Effective Way
- 2 Avoid the Software Overhead
- 3 Minimal Code Modifications

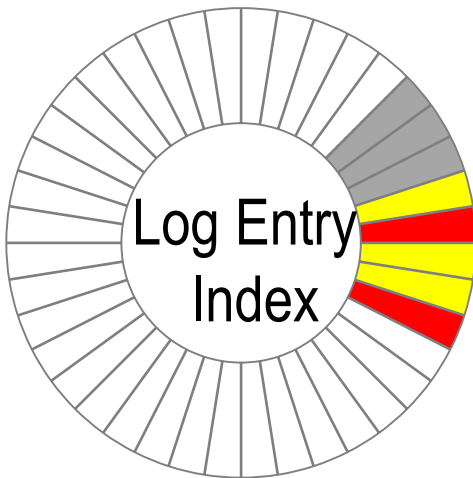
# Log Management in NVLogging

---

DRAM

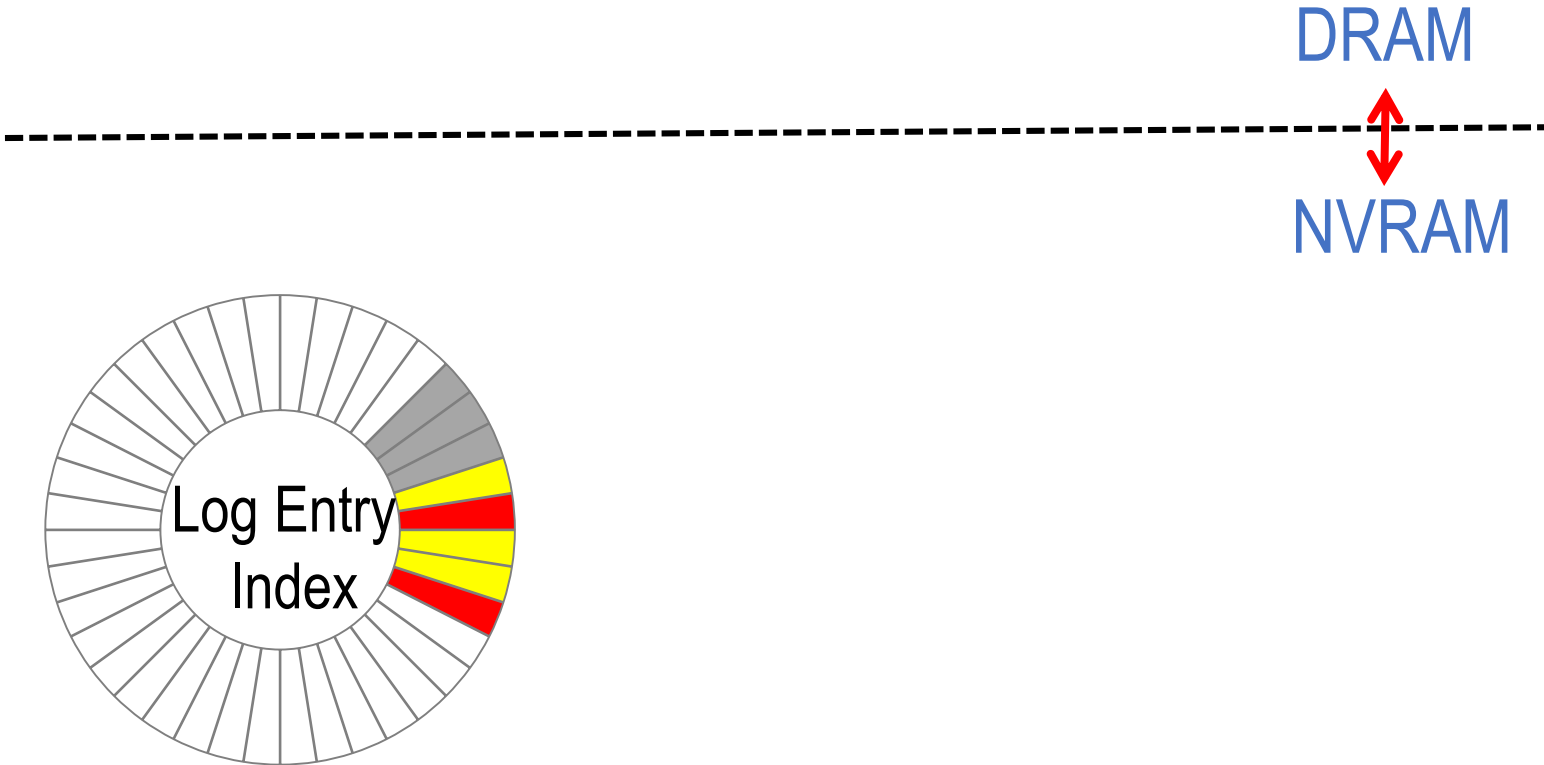


NVRAM



# Log Management in NVLogging

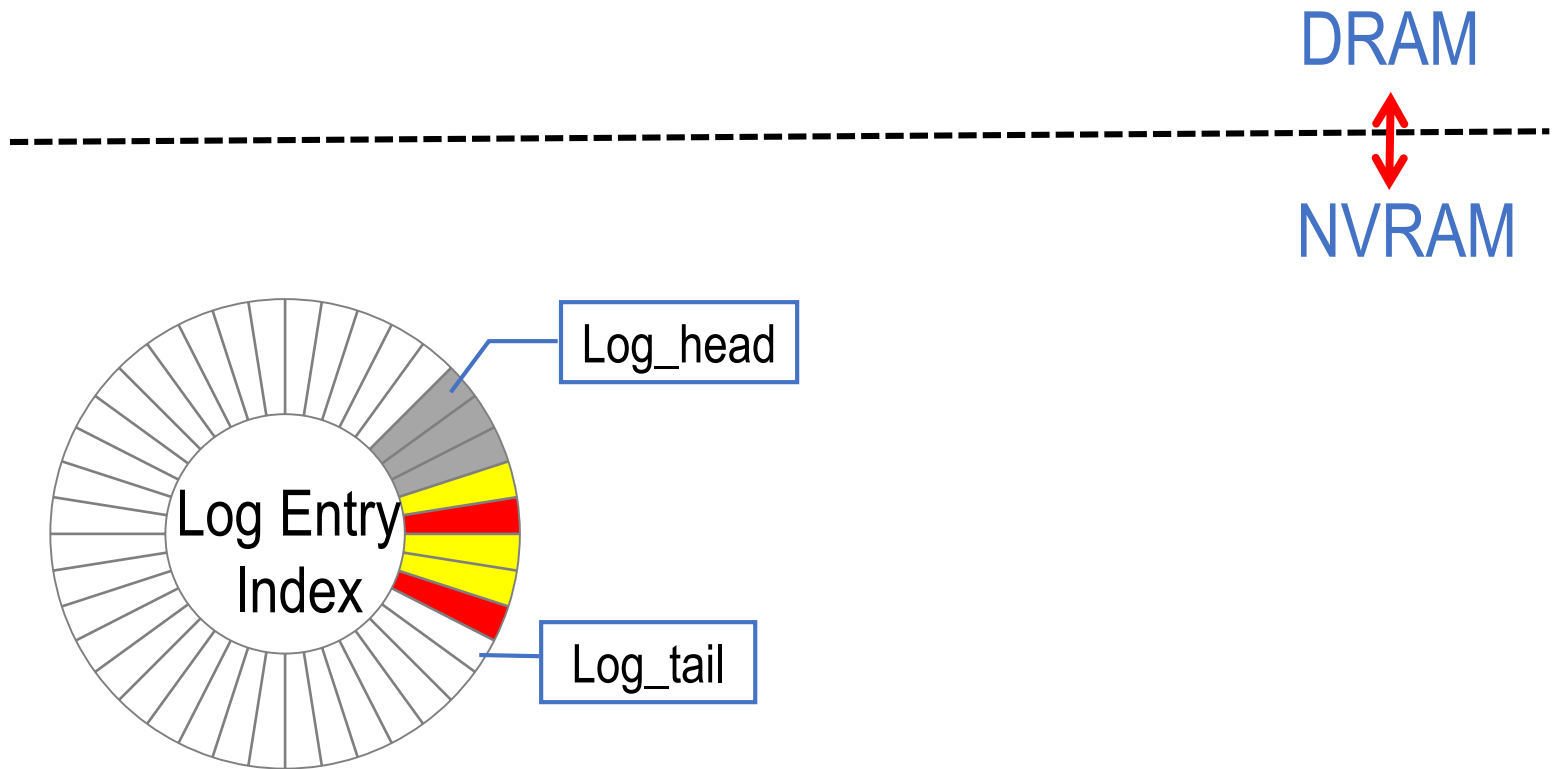
---





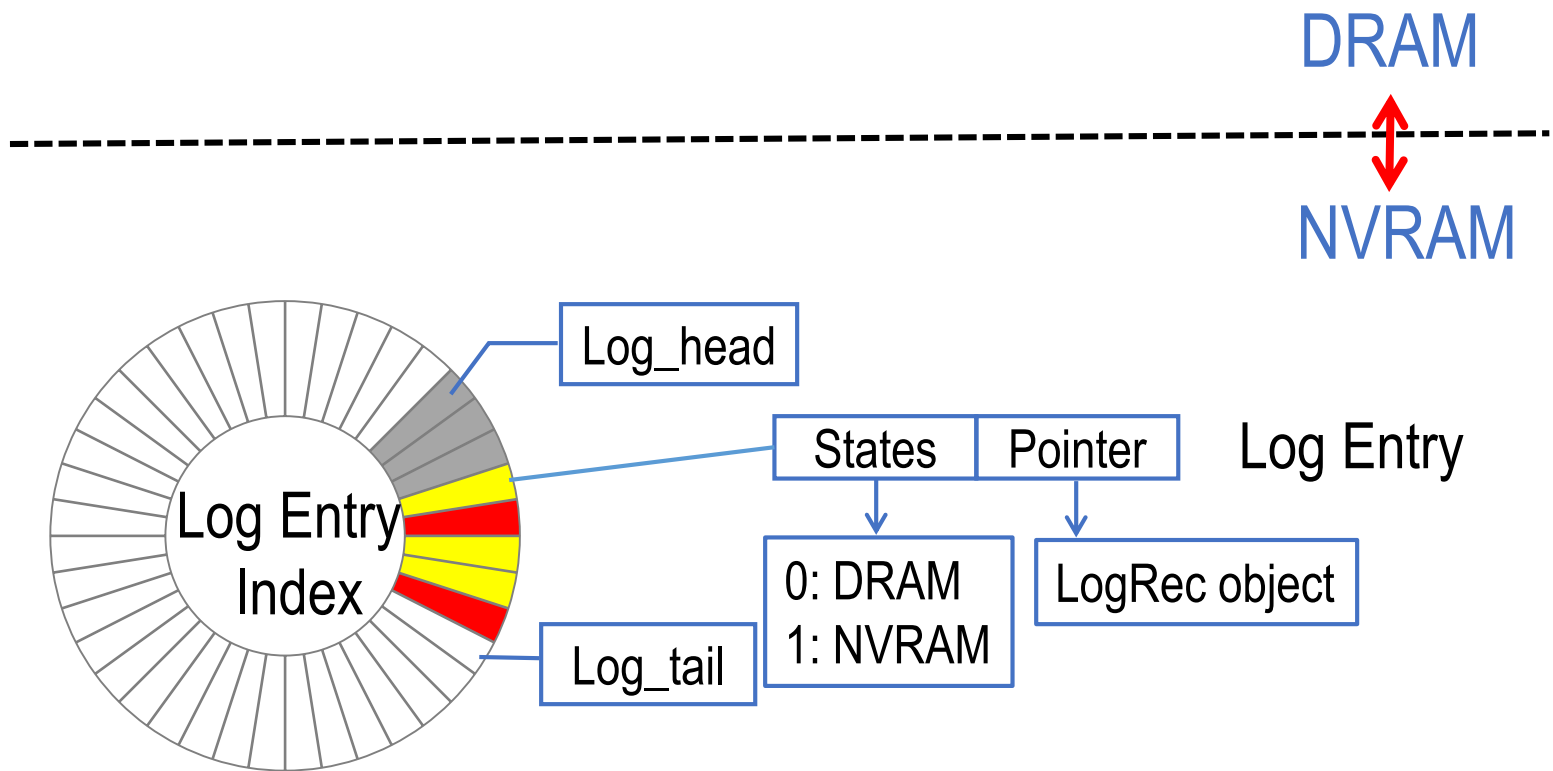
# Log Management in NVLogging

---



# Log Management in NVLogging

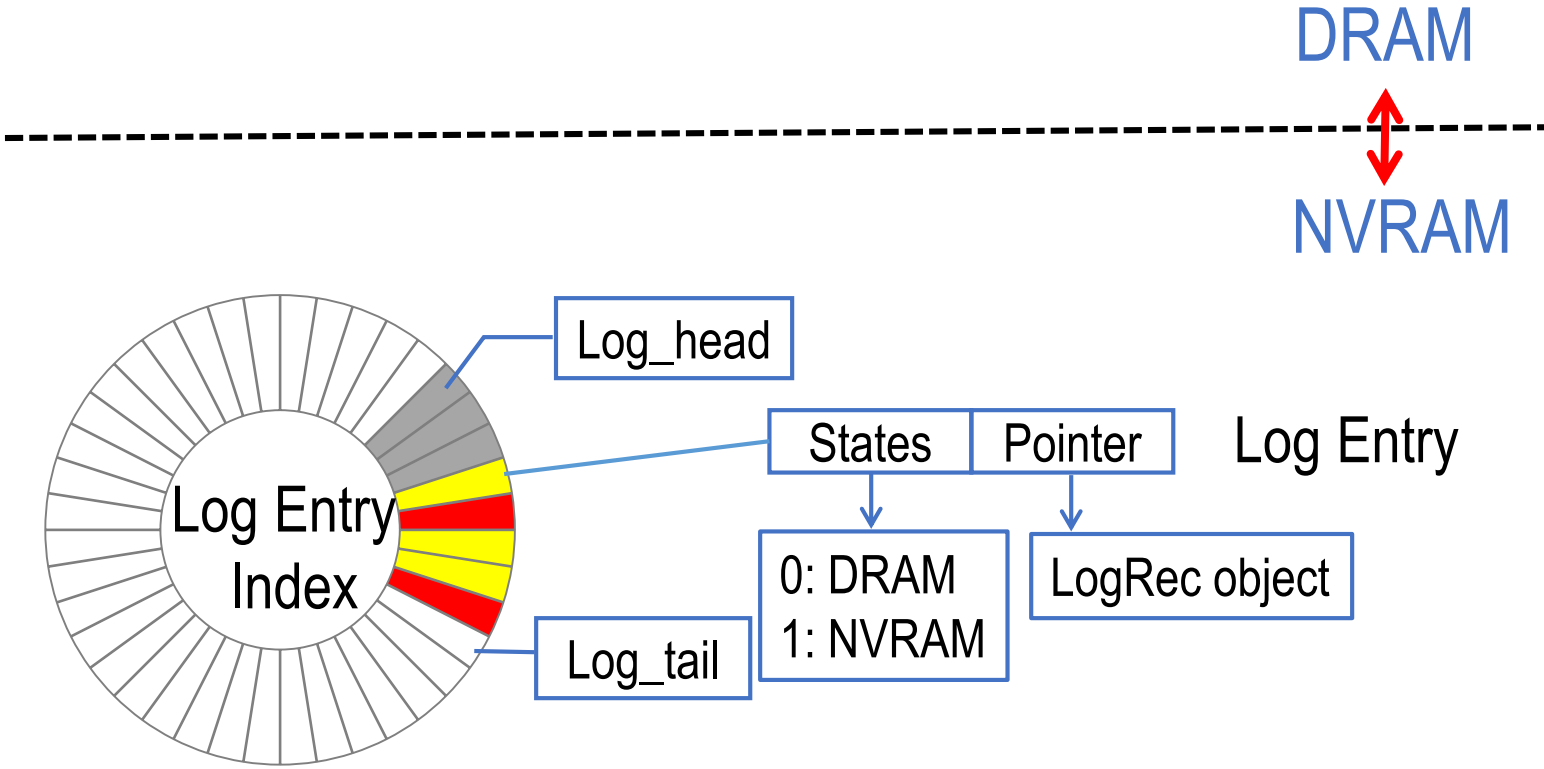
---



# Log Management in NVLogging

---

LogRec



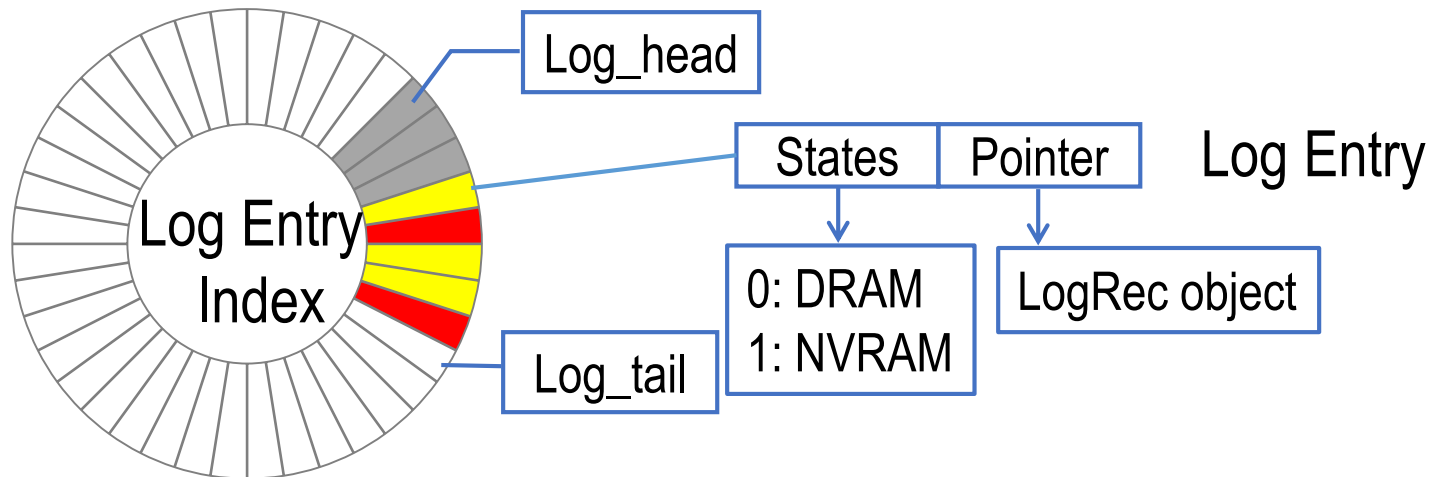
# Log Management in NVLogging

LogRec

Processor cache and DRAM are volatile.

DRAM

NVRAM



# Log Management in NVLogging

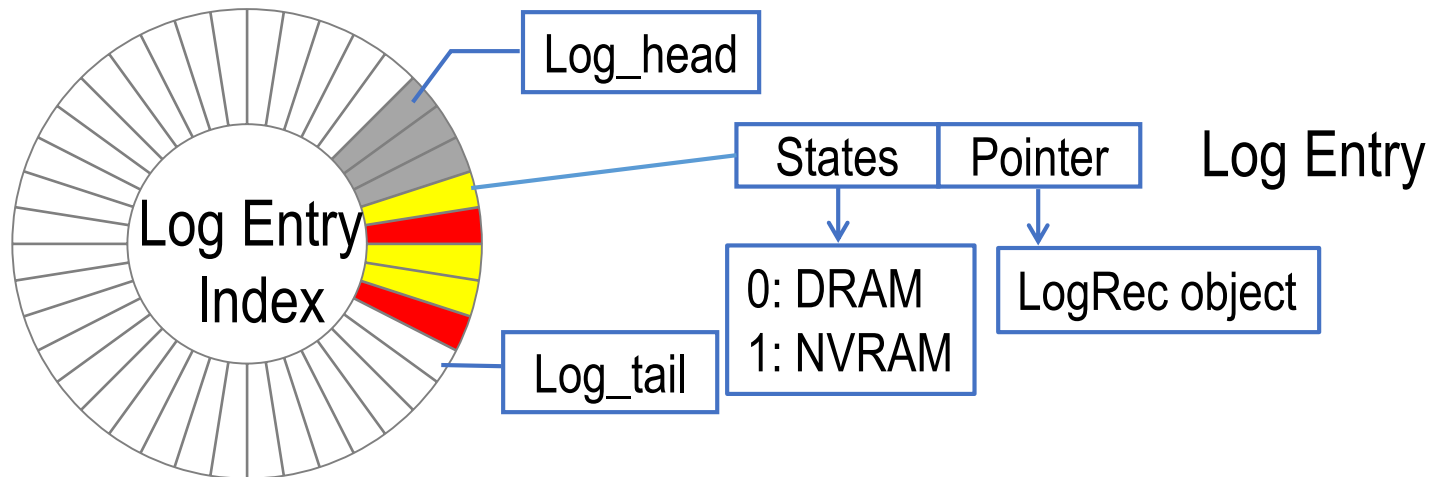
LogRec

Log persistence:  
flush-on-insert & flush-on-commit

DRAM

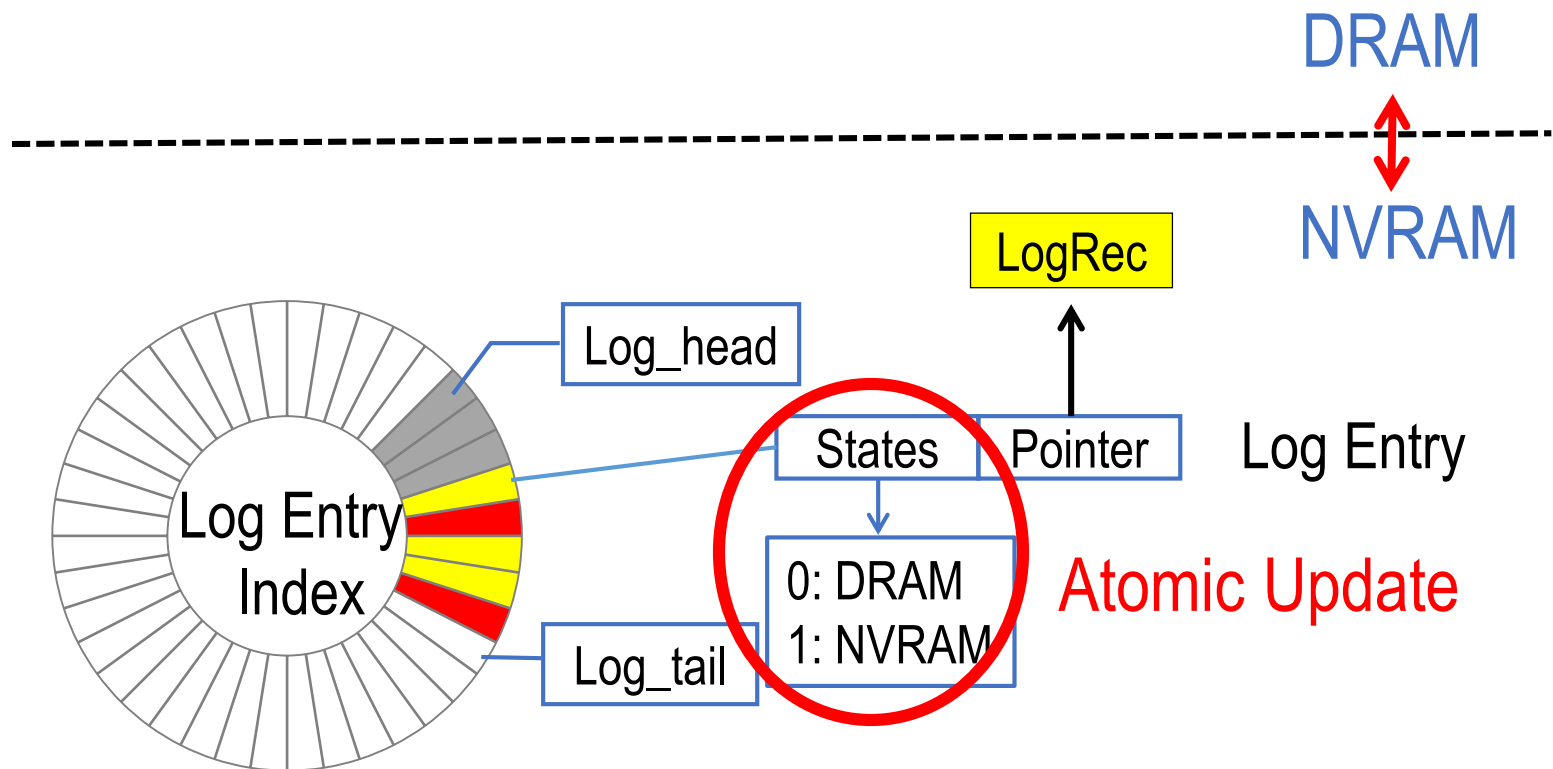


NVRAM



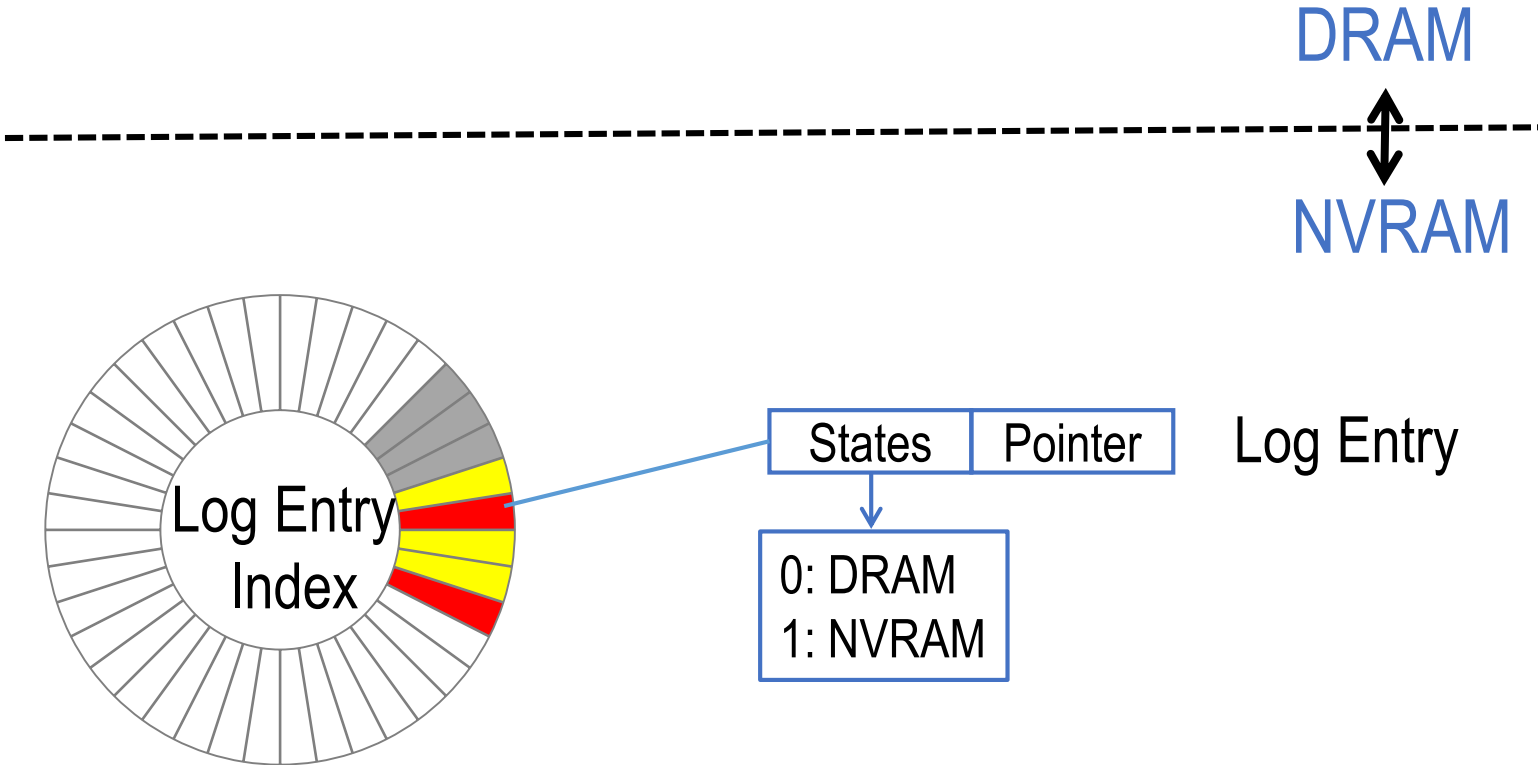
# Log Management in NVLogging

Log persistence:  
flush-on-insert & flush-on-commit

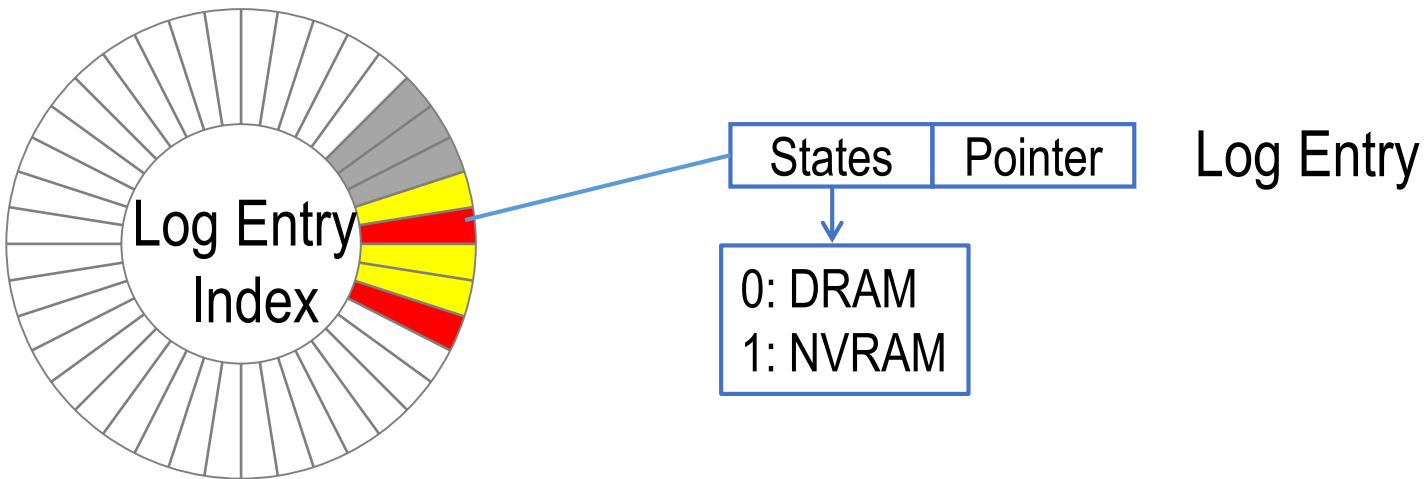
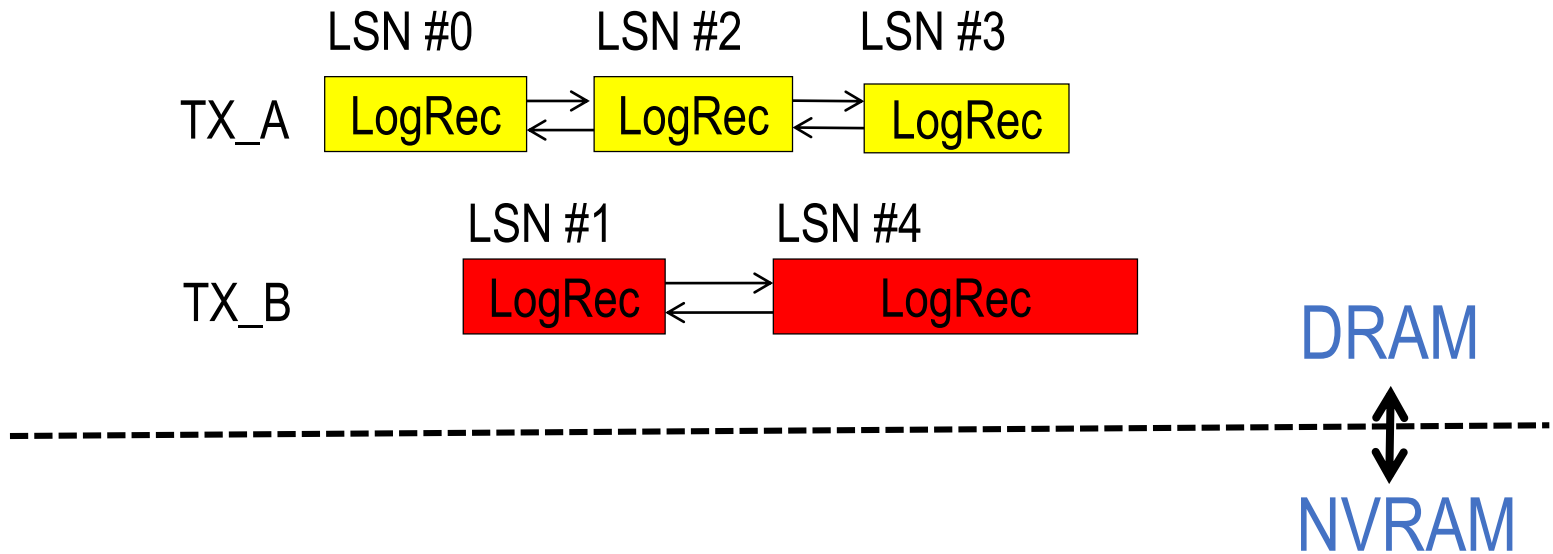


# Avoid Log Contention with Per-Transaction Logging

---

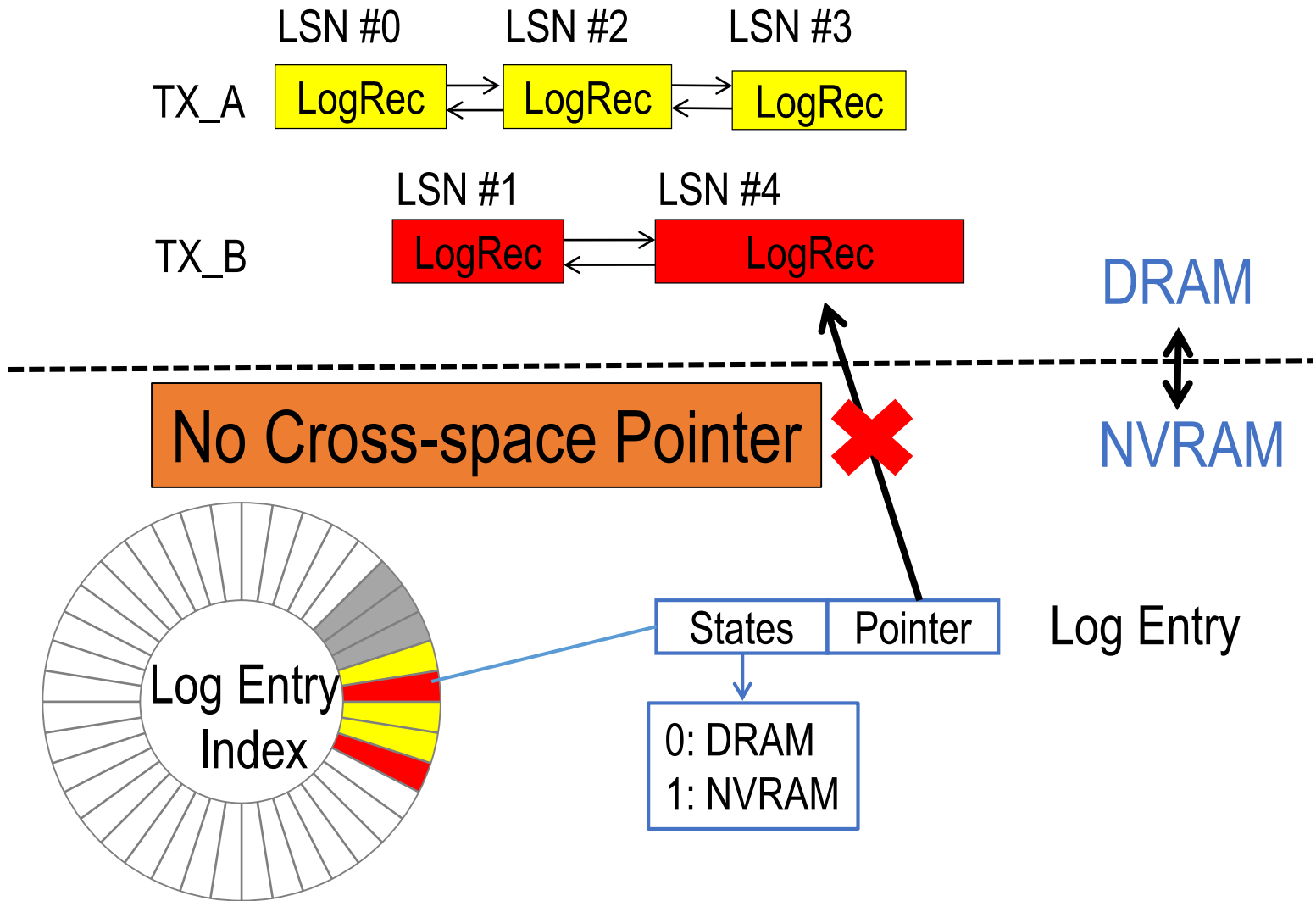


# Avoid Log Contention with Per-Transaction Logging

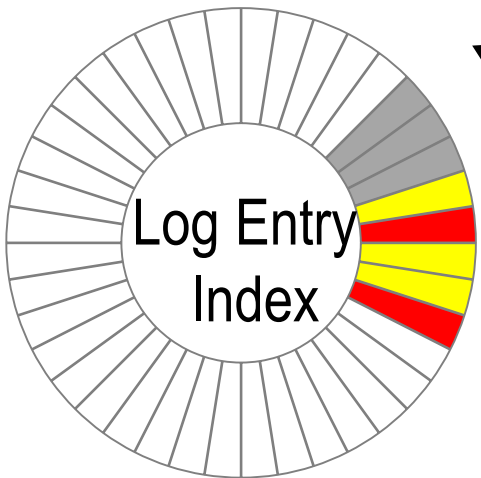
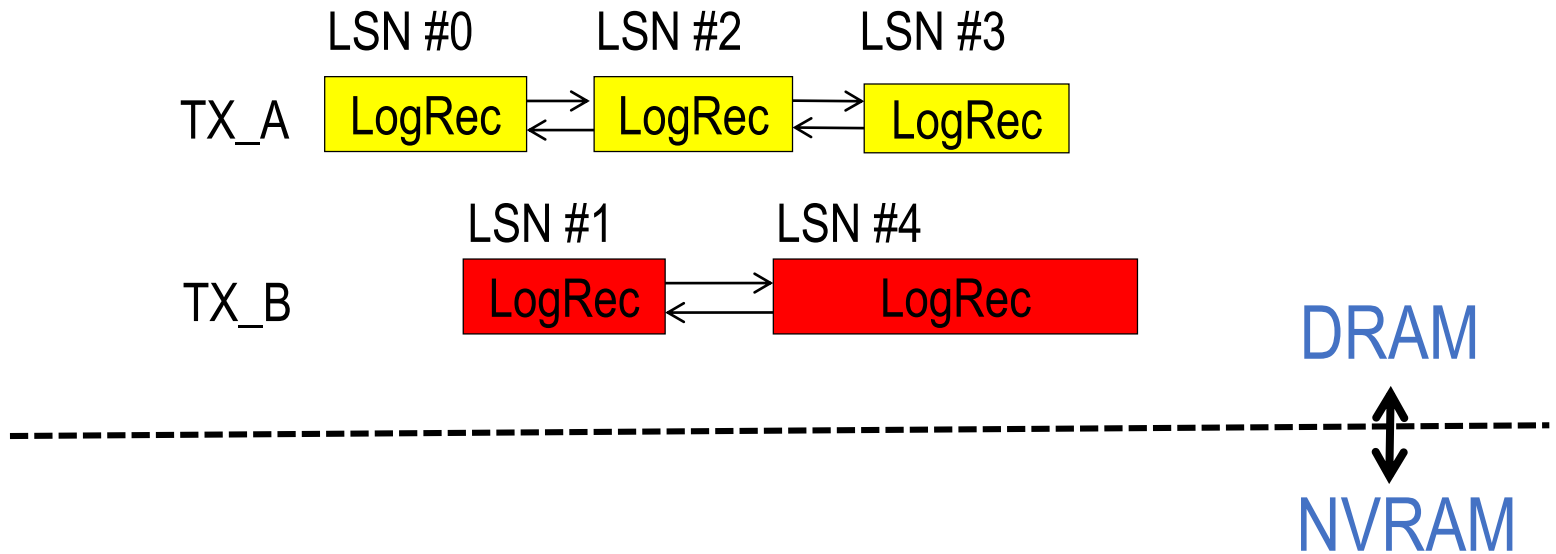




# Avoid Log Contention with Per-Transaction Logging



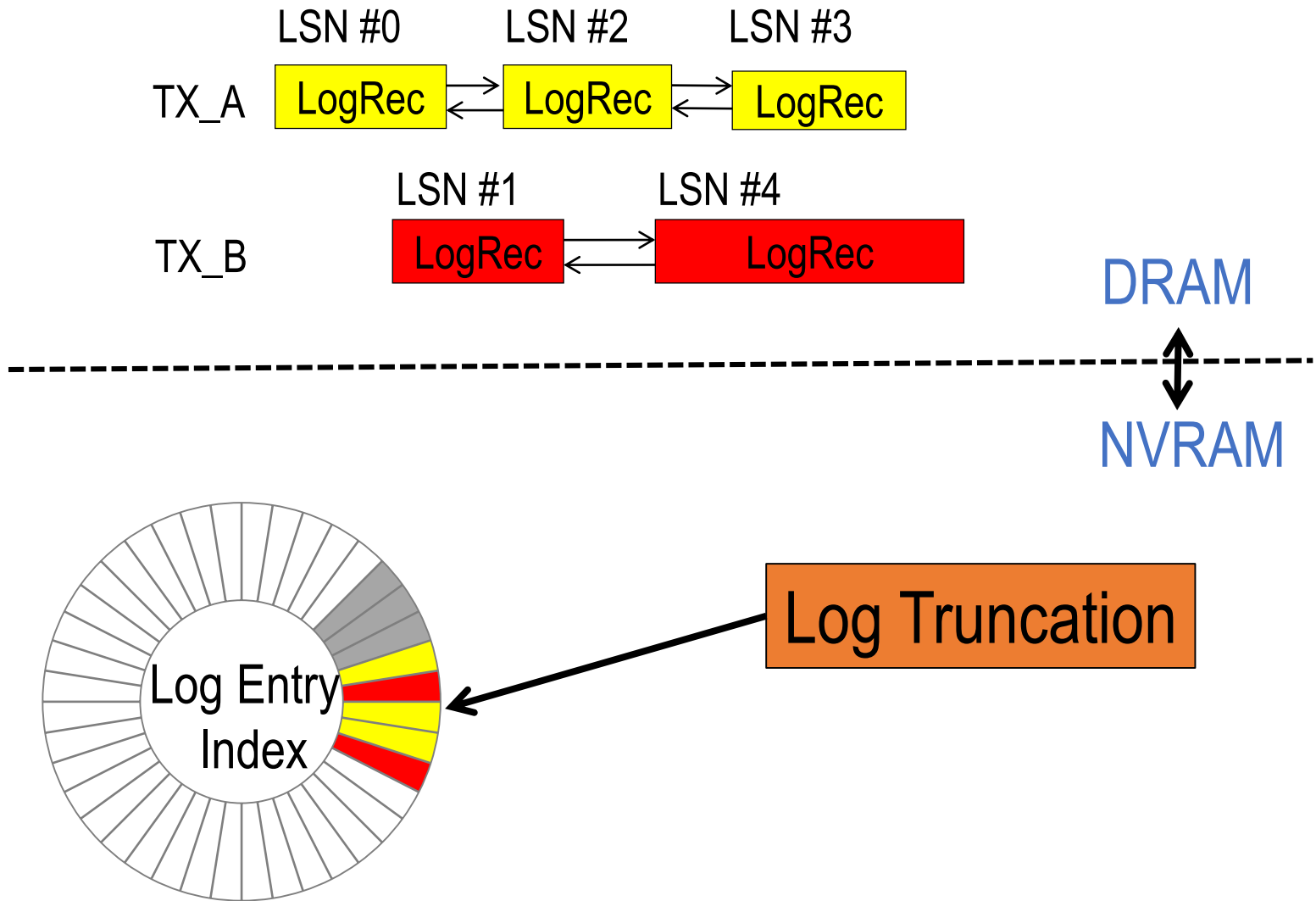
# Avoid Log Contention with Per-Transaction Logging



Recovery

Using back pointer to access log objects

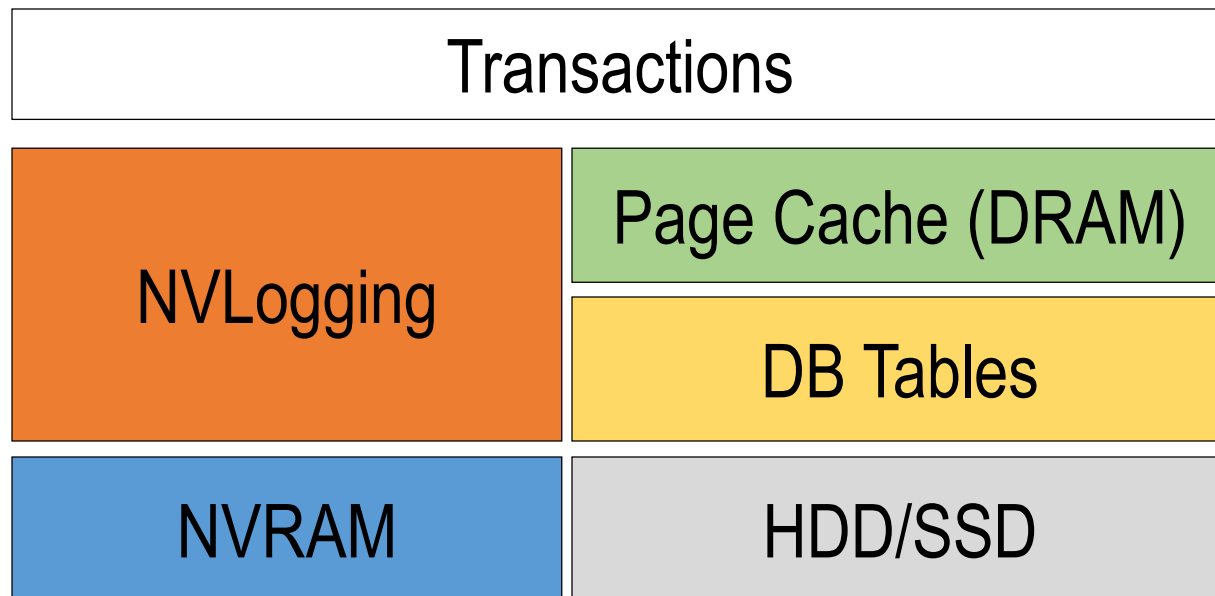
# Avoid Log Contention with Per-Transaction Logging



# Implementation and Experimental Setup

---

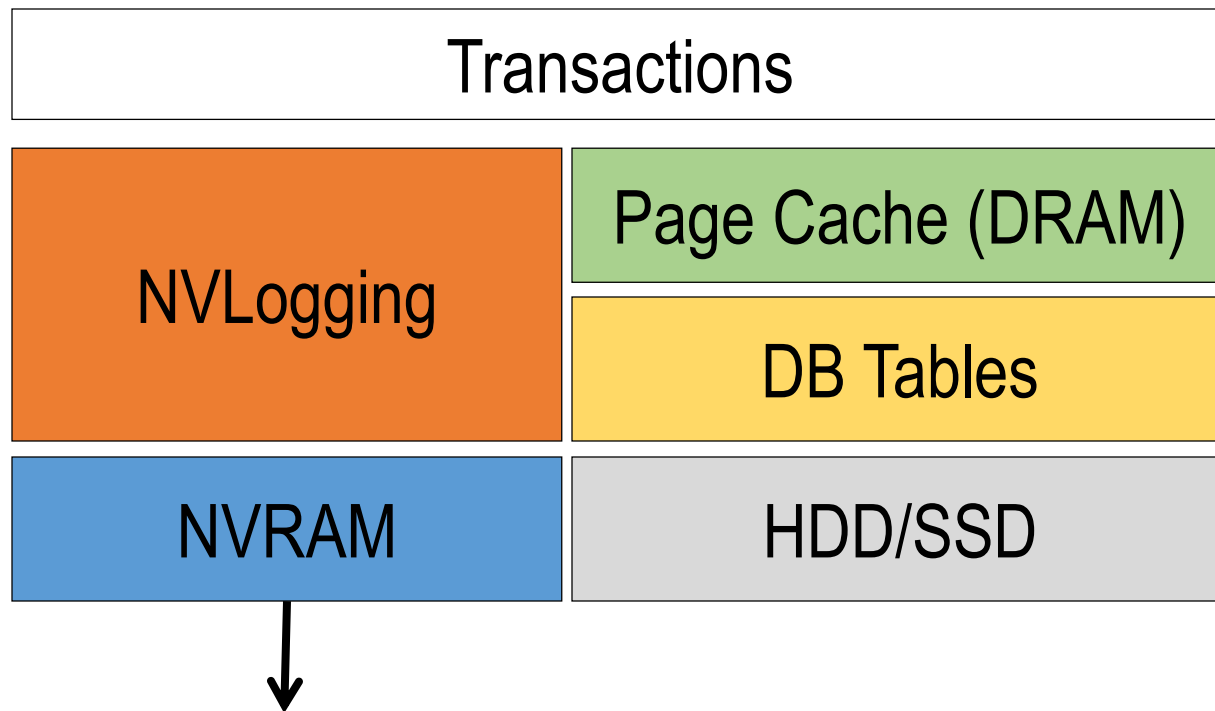
## Shore-MT + Shore-Kits Benchmark



# Implementation and Experimental Setup

---

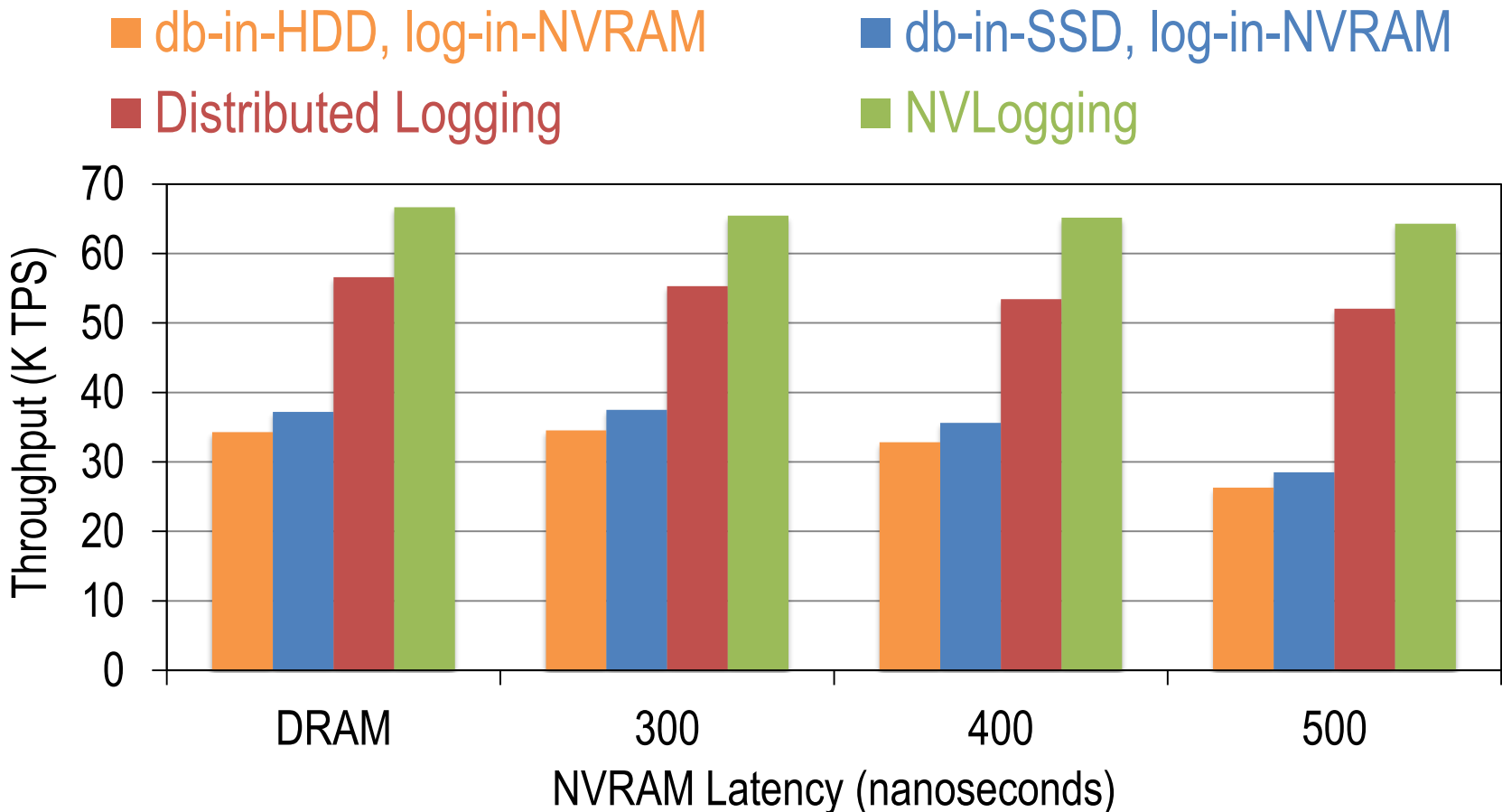
## Shore-MT + Shore-Kits Benchmark



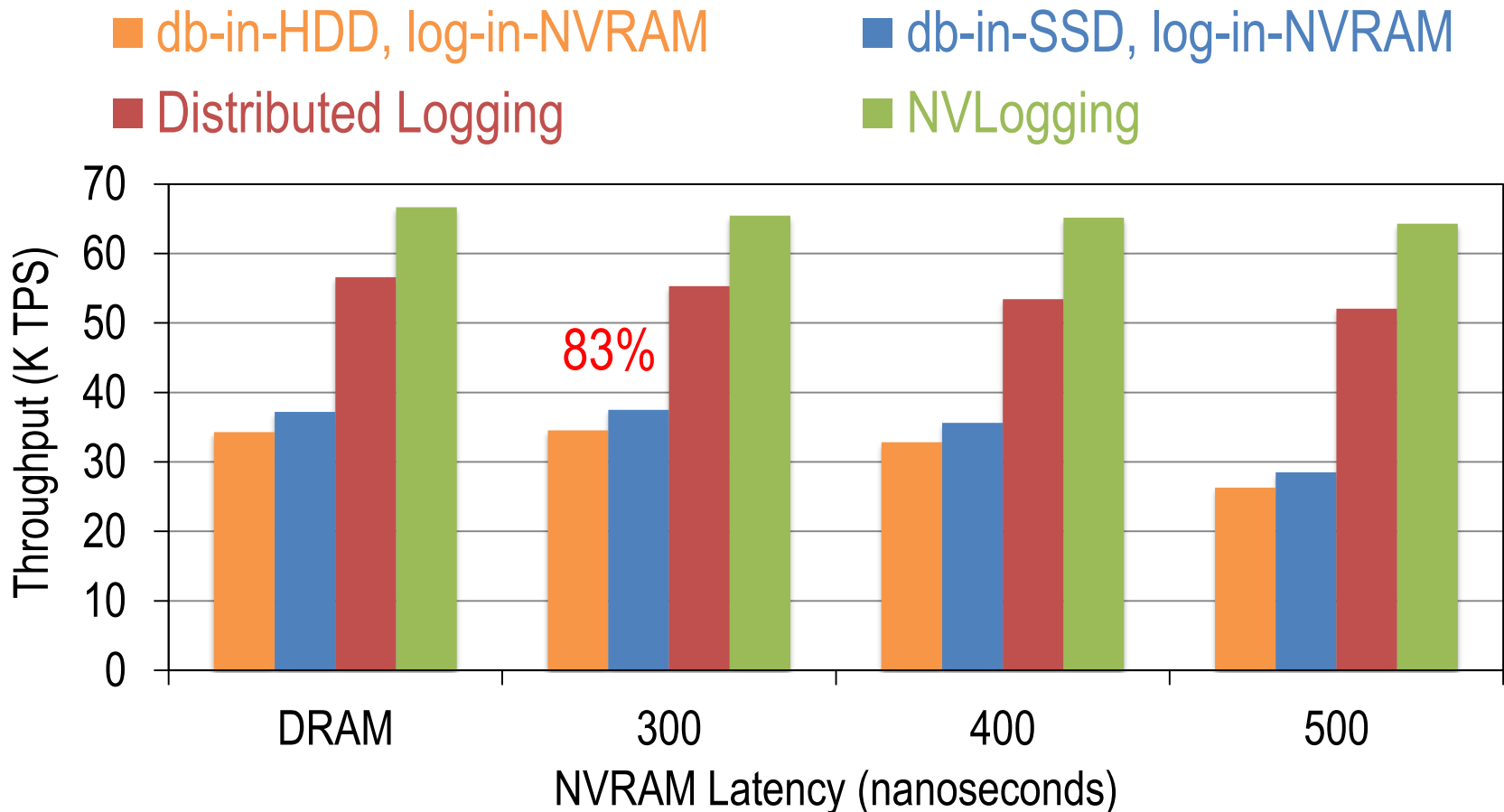
Intel Persistent Memory Emulator Platform

Xeon-EP @ 2.6 GHz, 64 GB DRAM + 256 GB NVRAM

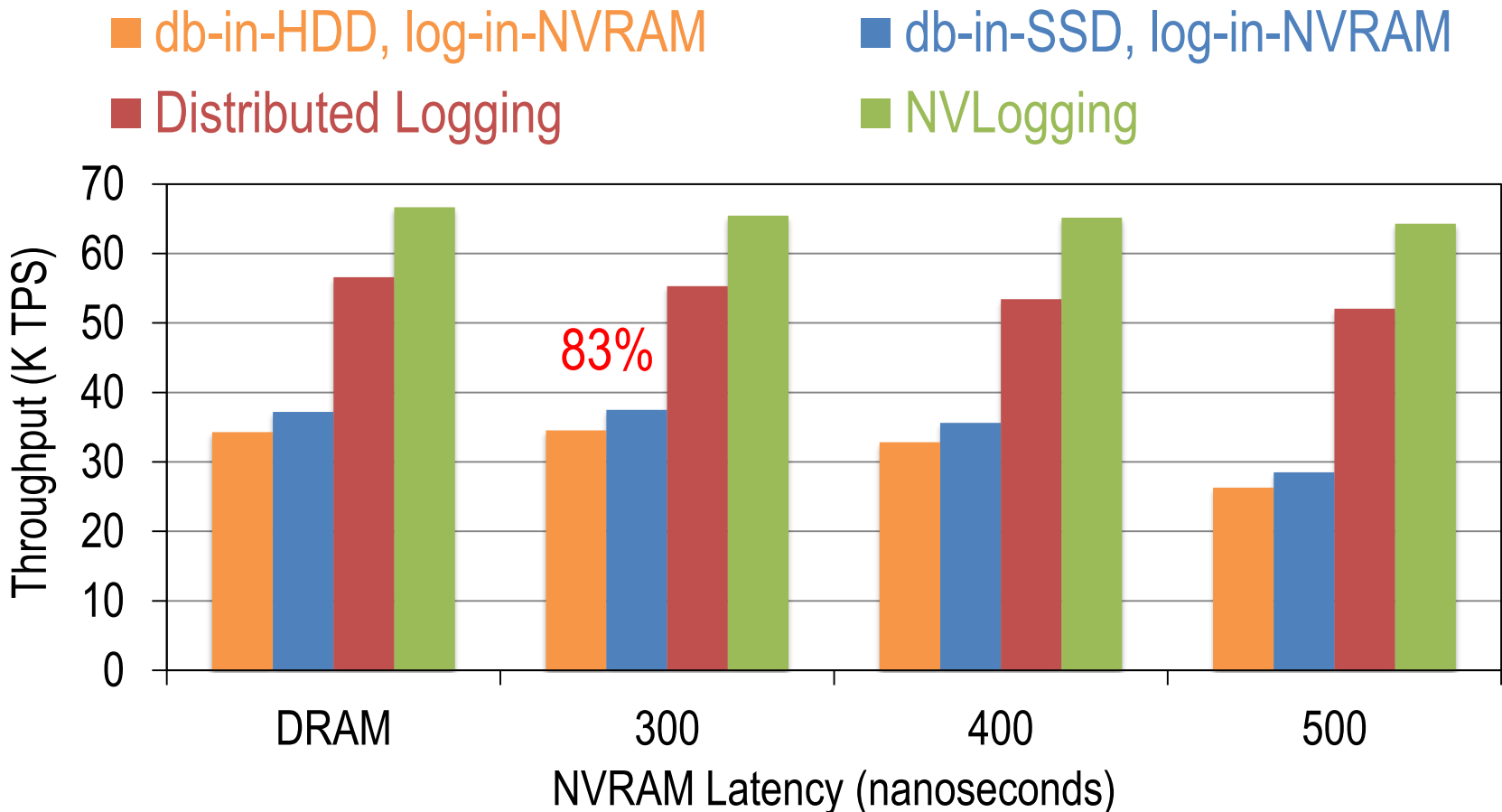
# Benefits on Throughput with TPCB



# Benefits on Throughput with TPCB



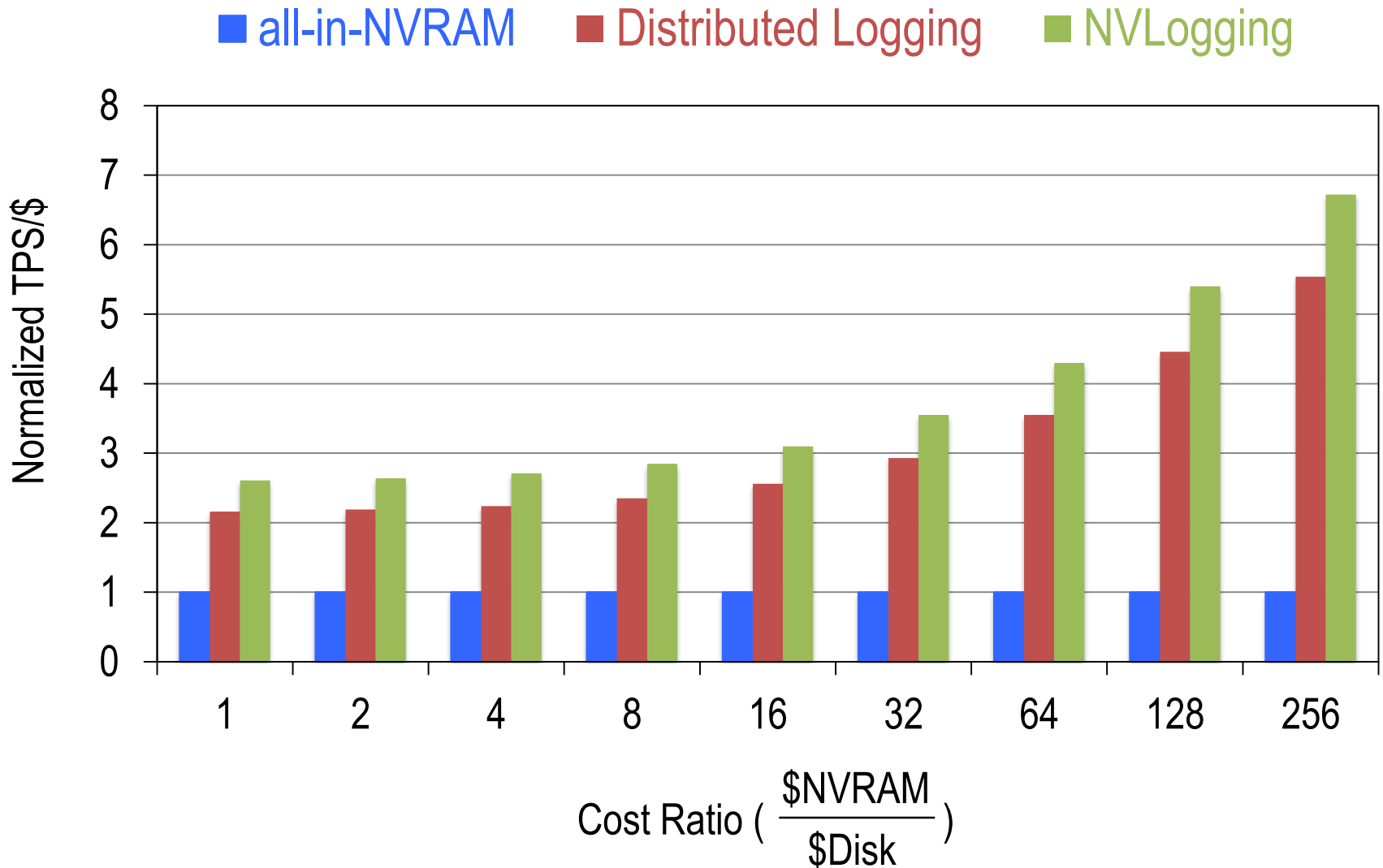
# Benefits on Throughput with TPCB



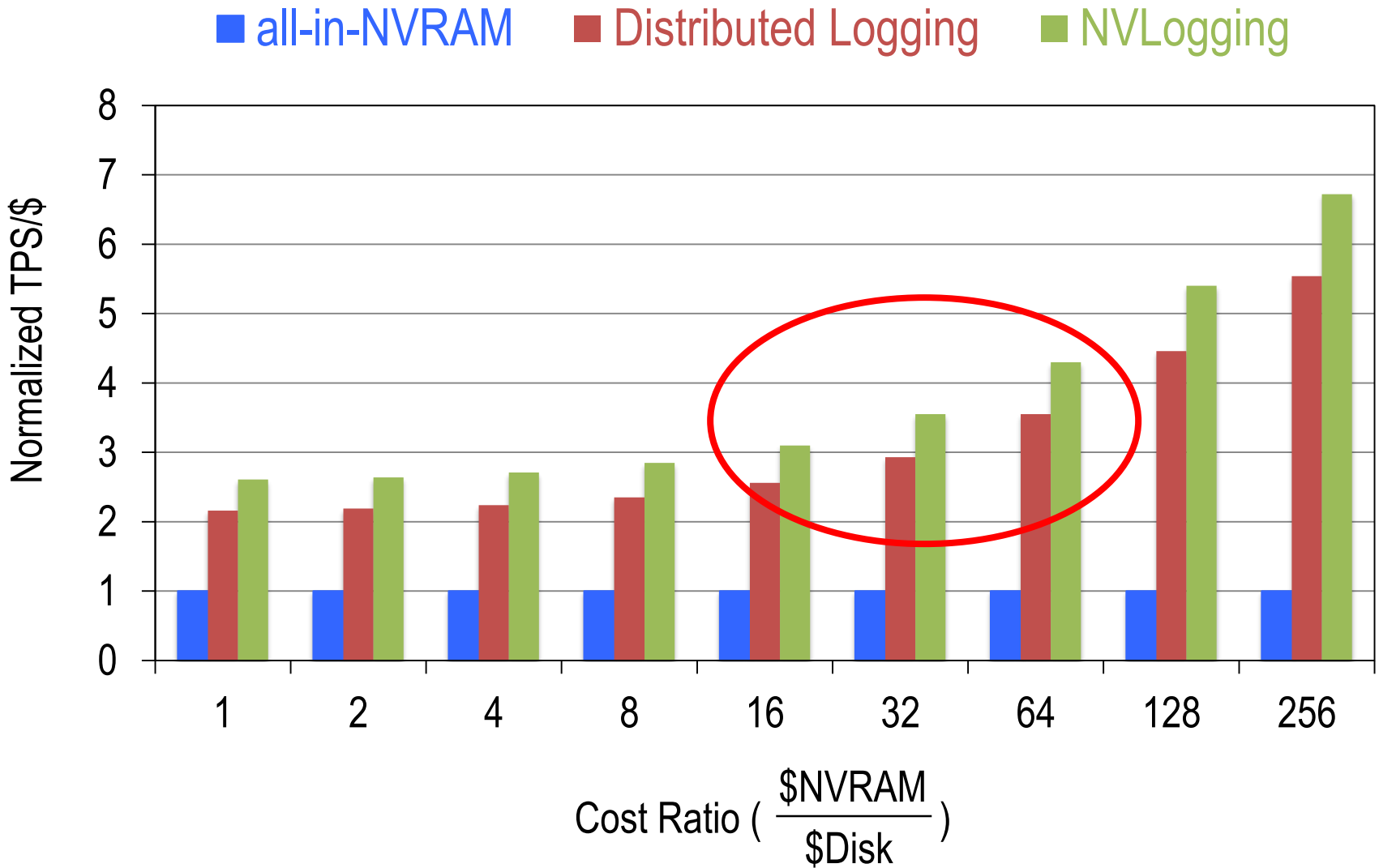
NVLogging improves TPS by 2.3x,  
compared to the baseline (db-in-SSD, log-in-NVRAM)



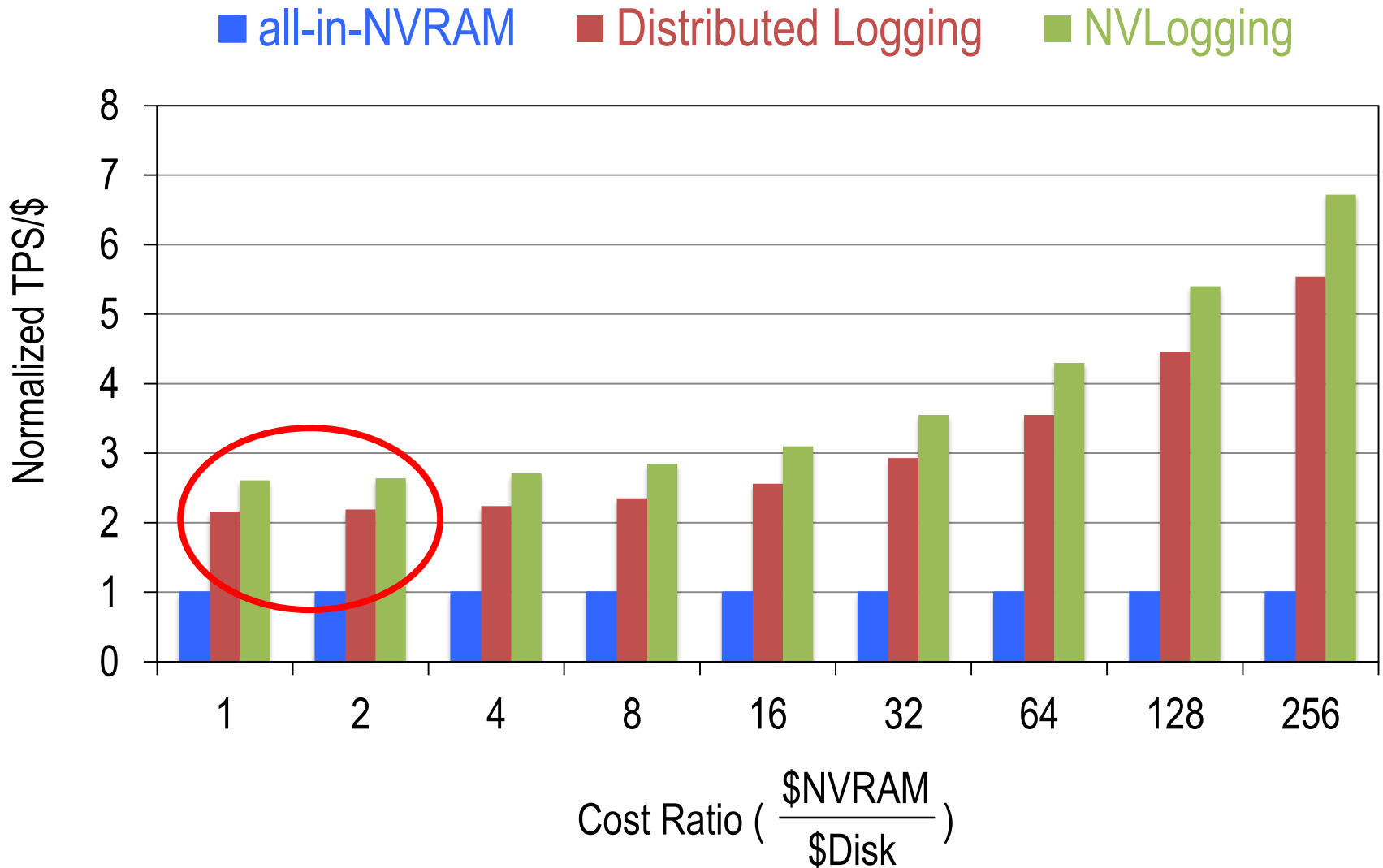
# Benefits on Cost Savings



# Benefits on Cost Savings

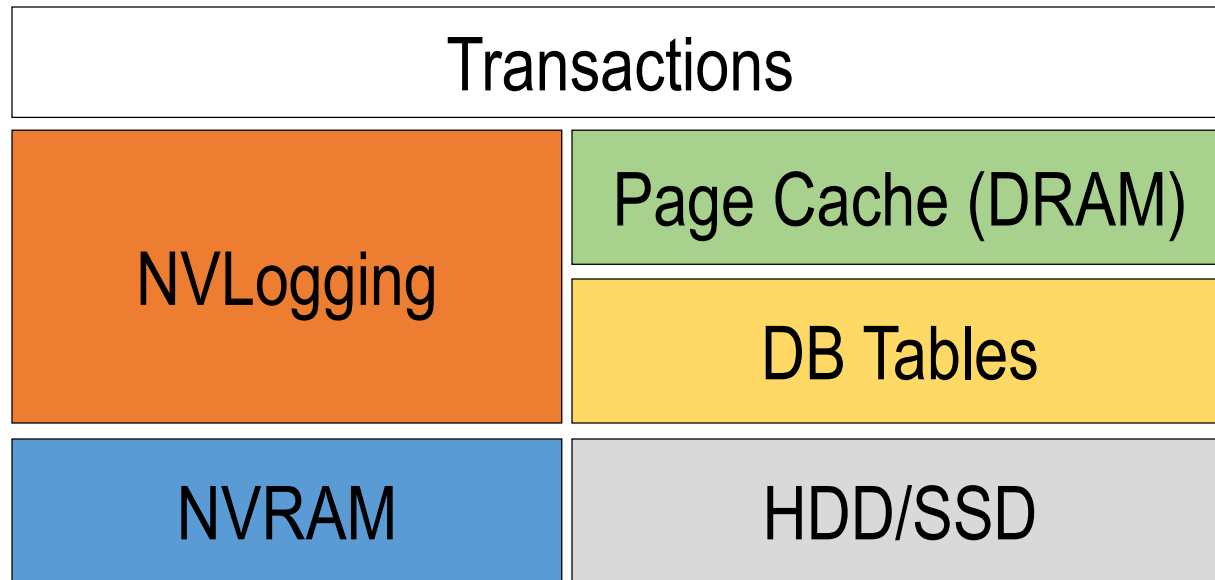


# Benefits on Cost Savings



# Conclusion

---



## 1 Cost-effective Solution for DB system

Higher TPS/\$

## 2 Per-transaction Logging with NVRAM

Exploit NVRAM's both non-volatility and byte-addressability

# Thanks!

---

**Jian Huang**

[jian.huang@gatech.edu](mailto:jian.huang@gatech.edu)

Karsten Schwan

Moinuddin K. Qureshi



---

# Q&A