Hardware-driven Undo+Redo Logging

Jishen Zhao

https://users.soe.ucsc.edu/~jzhao/

Computer Engineering, UC Santa Cruz

October 11, 2017

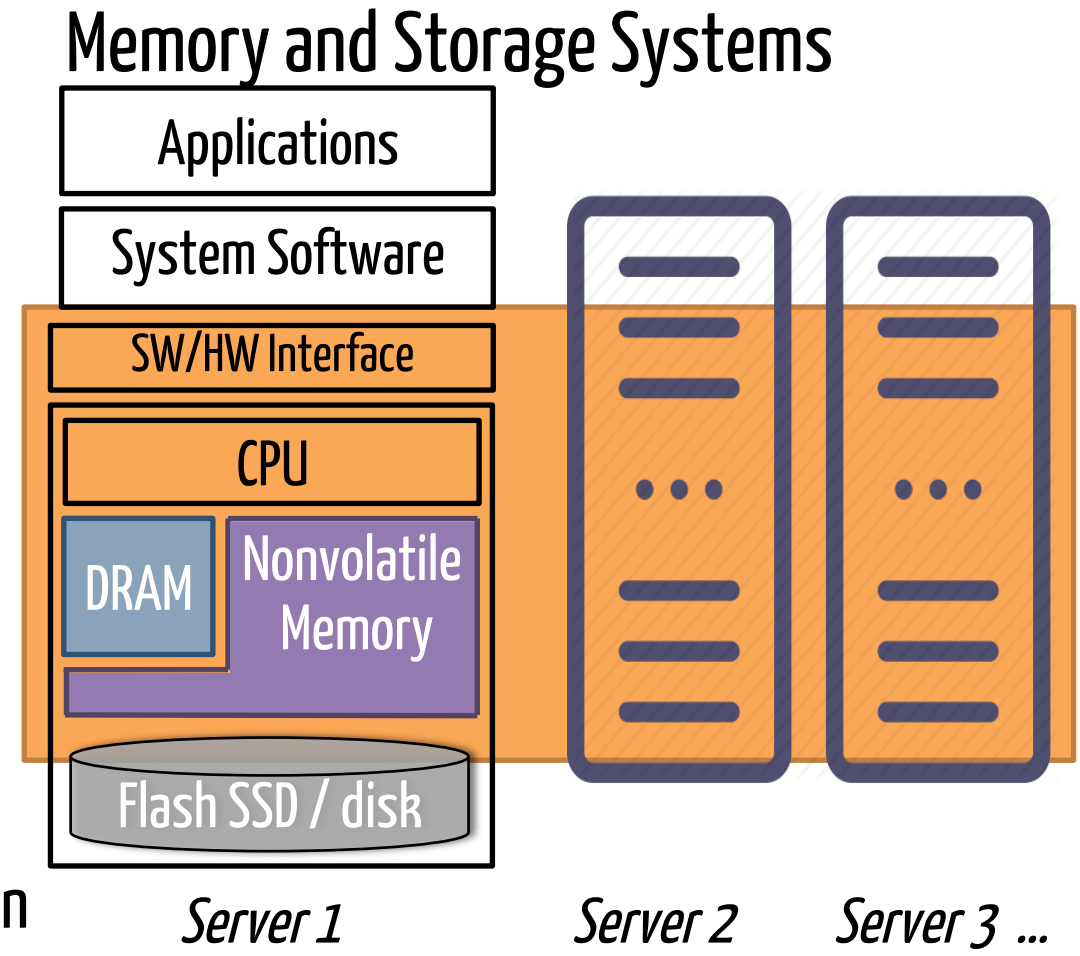
HPTS
HIGH PERFORMANCE TRANSACTION SYSTEMS

Research Overview

- **Main research topics**

Processing In Memory | Persistent Memory | Memory Network

- **Research efforts**
  - Workload characterization
  - Software/hardware cooperative mechanisms
  - Software/hardware interface design
  - Hardware-based accelerators



Typical memory and storage hierarchy:

Memory Fast access to working data

Storage Data persistence

Persistent memory:

Fast memory interface + persistence

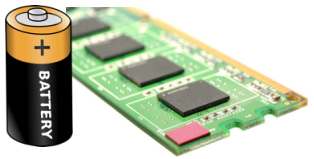
Dry it

Persistent memory is ~~coming!~~ here!

Hardware – Nonvolatile random-access memories (NVRAMs)

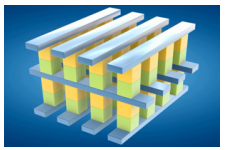
Not flash memory…

Battery-backed DRAM

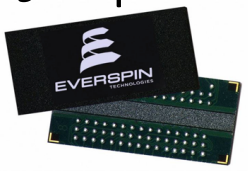


NV-DIMM

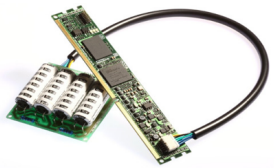
3D Xpoint



DDR3 Compatible MRAM



DRAM w/ Ultra-capacitor



Software – Persistent-memory-aware system software

Persistent Memory Support in OS

Using DAX in Windows

DAX Volume Creation

→ Format n: /dax /q

→ Format-Volume -DriveLetter n -

DAX Volume Identification

Is it a DAX volume?

→ call GetVolumeInformation("C:\", ...)

→ check lpFileSystemFlags for FILE_DAX_VOLUME (0x20000000)

Microsoft

NVM.PM.FILE ACTIONS

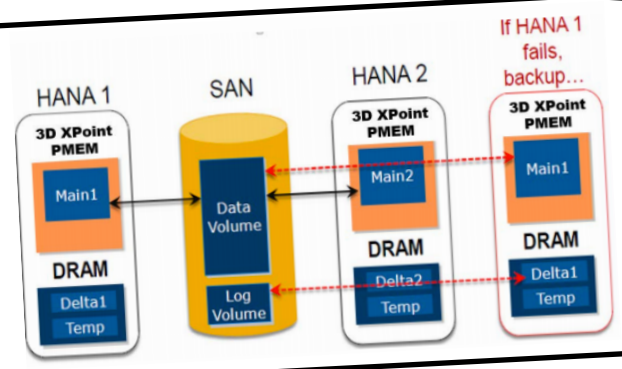
- Implemented by kernel
- Implemented in userspace
  - Optimized_flush
  - Optimized_flush_and_verify

redhat.

Persistent Memory File Systems

File system	Metadata atomicity	Data atomicity	Mmap Atomicity [1]
BPFS	Yes	Yes [2]	No
PMFS	Yes	No	No
Ext4-DAX	Yes	No	No
SCMFS	No	No	No
Aerie	Yes	No	No
NOVA	Yes	Yes	Yes

Persistent Memory Aware Database



PELOTON

CARNEGIE MELLON DATABASE GROUP

Persistent Memory Support Over Fabric

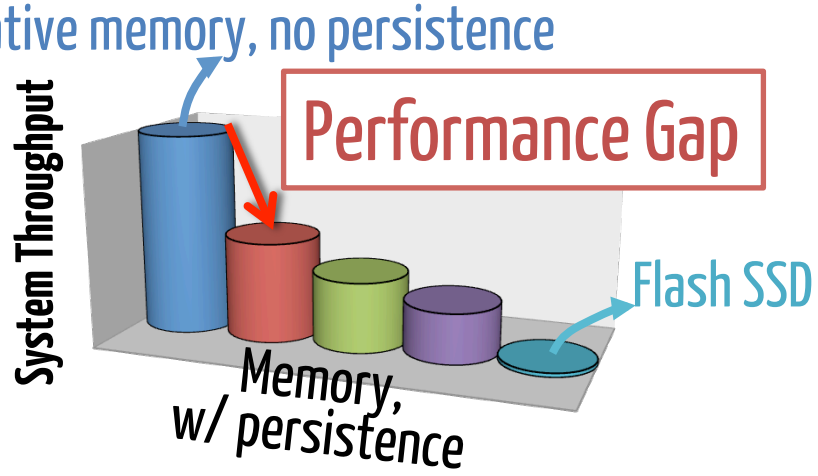
Persistent memory is ~~coming!~~ here!

...but unlocking its full potential isn't easy

Dry it with "Logs"

And copy-on-write, checkpointing, etc.

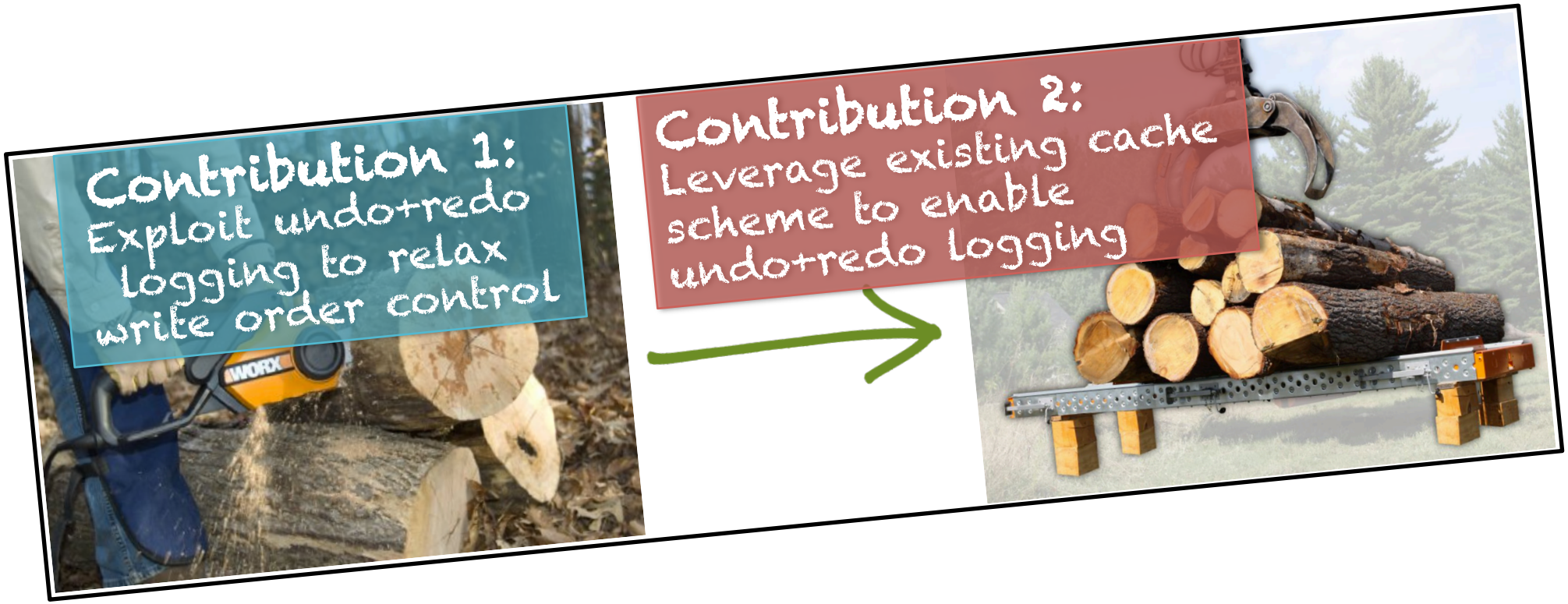
- Persistence
  - Used to be a property of storage systems
  - Now needs to be maintained in the memory system



[Zhao +, MICRO'13]

Opportunity

Hardware-driven undo + redo Logging

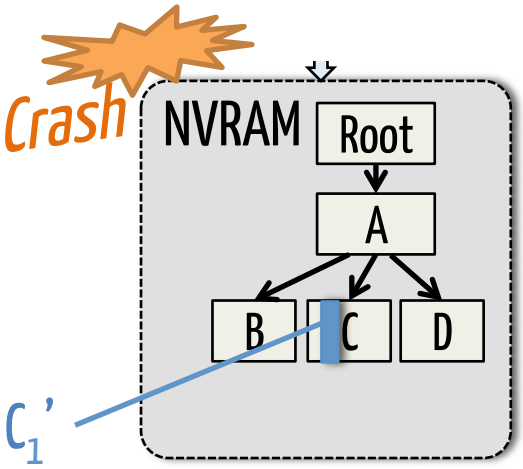


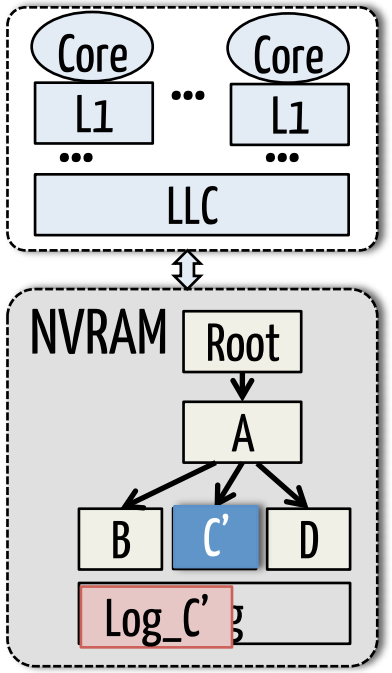
Persistence requirement in cache-memory hierarchy

Update persistent memory with a transaction

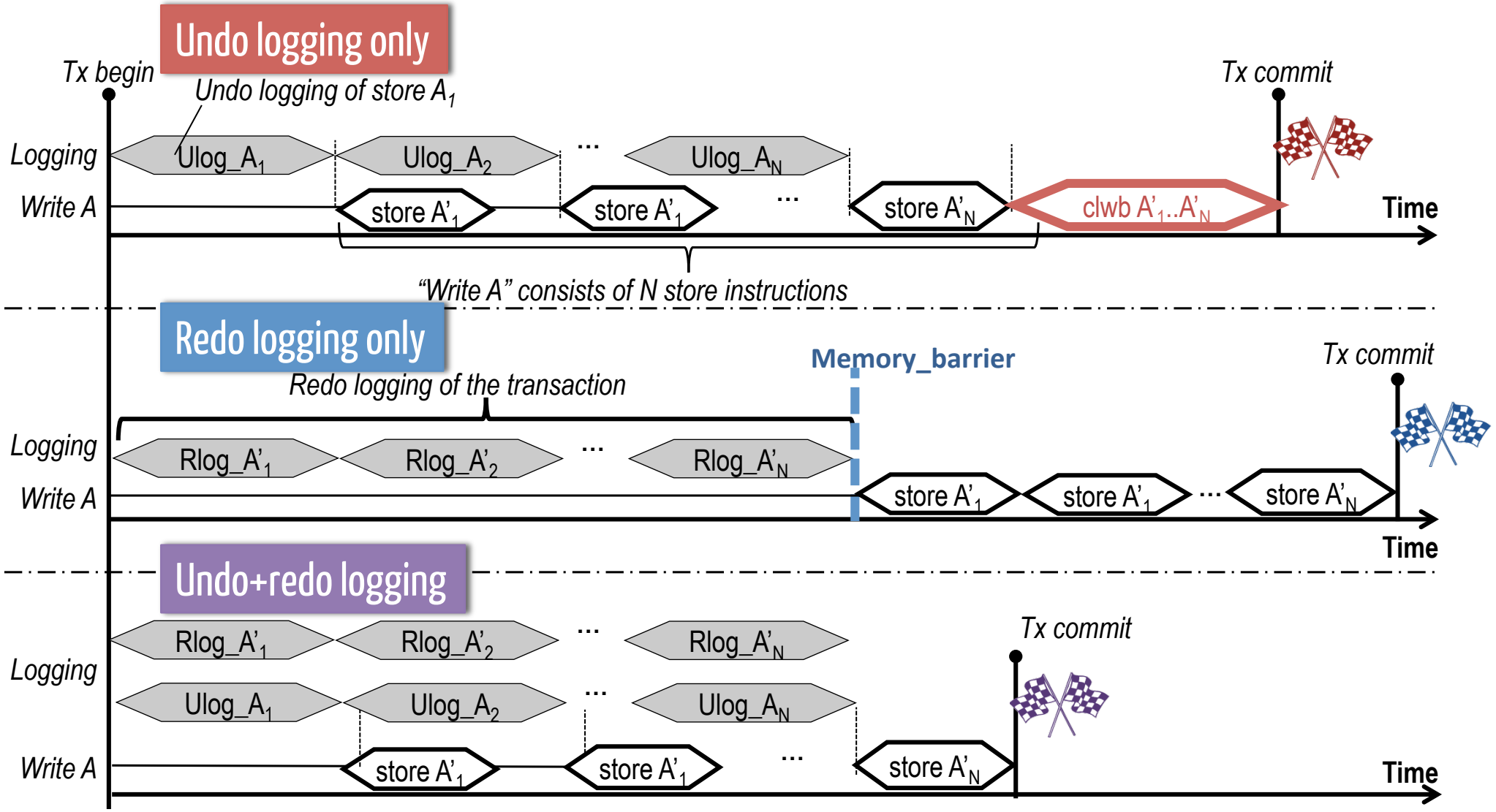
```
Tx_begin
   do some reads
   do some computation

   write C
Tx_commit
```

Micro-ops:
store C'_1
store C'_2
...

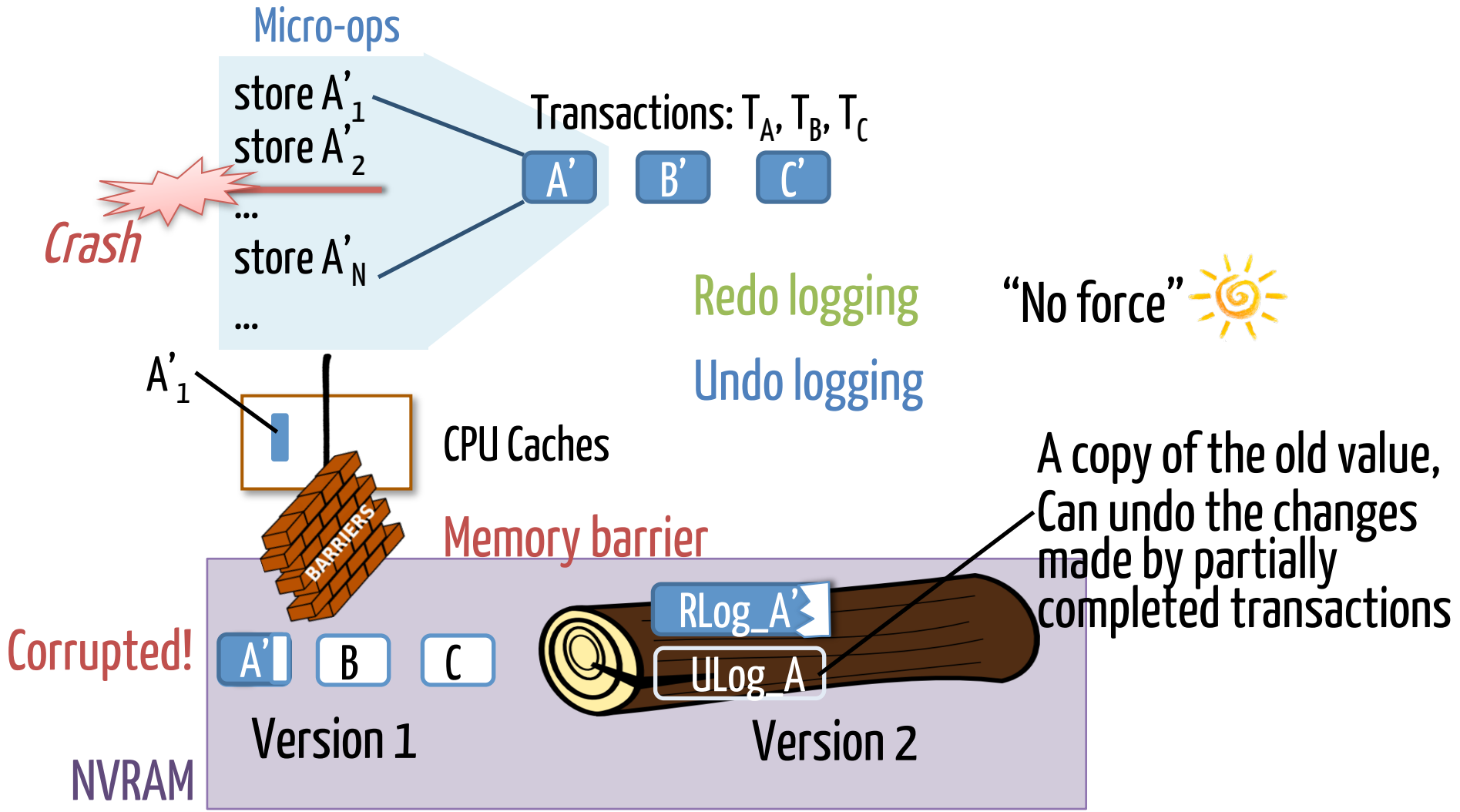




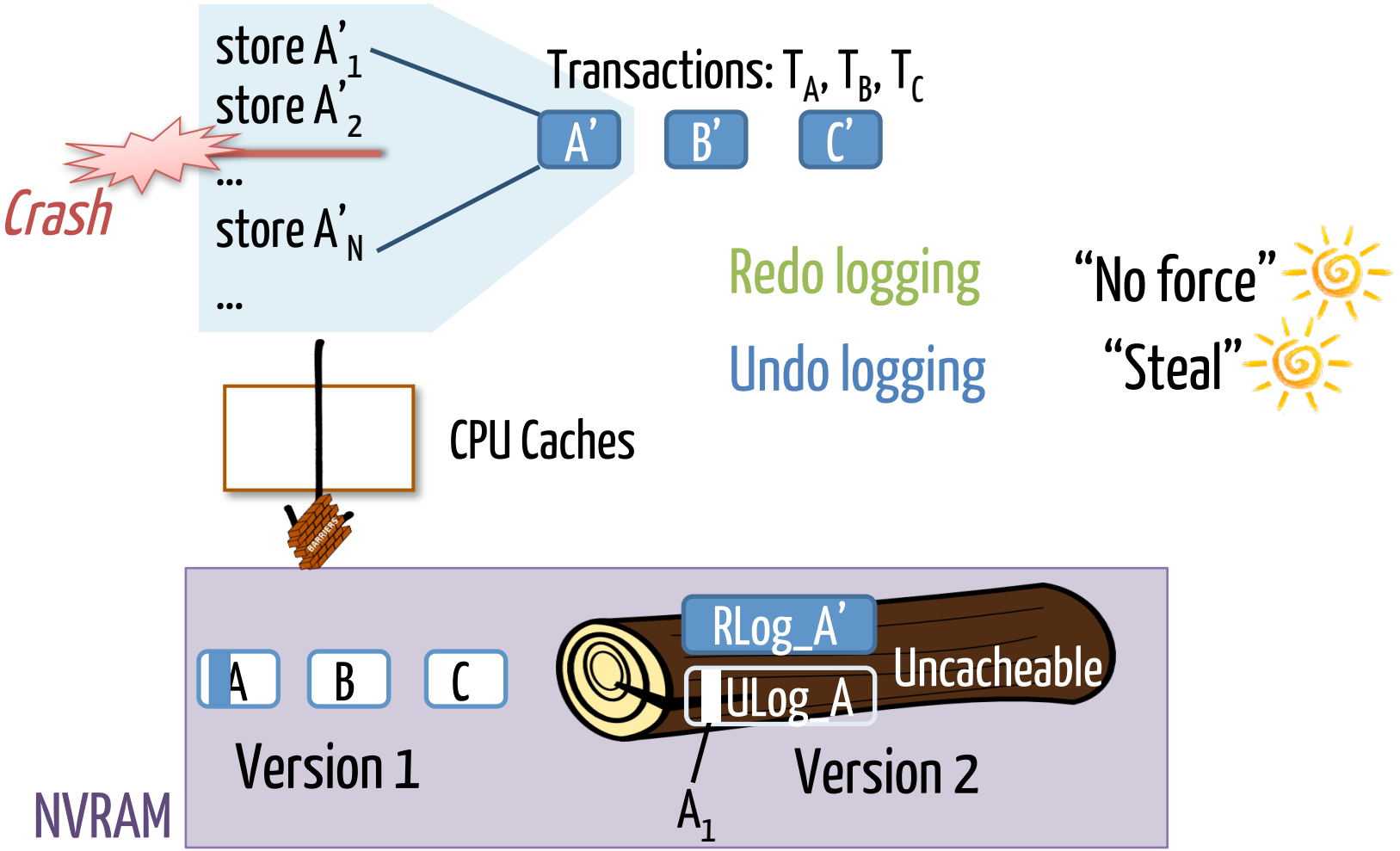
Preview of performance benefit with undo+redo logging



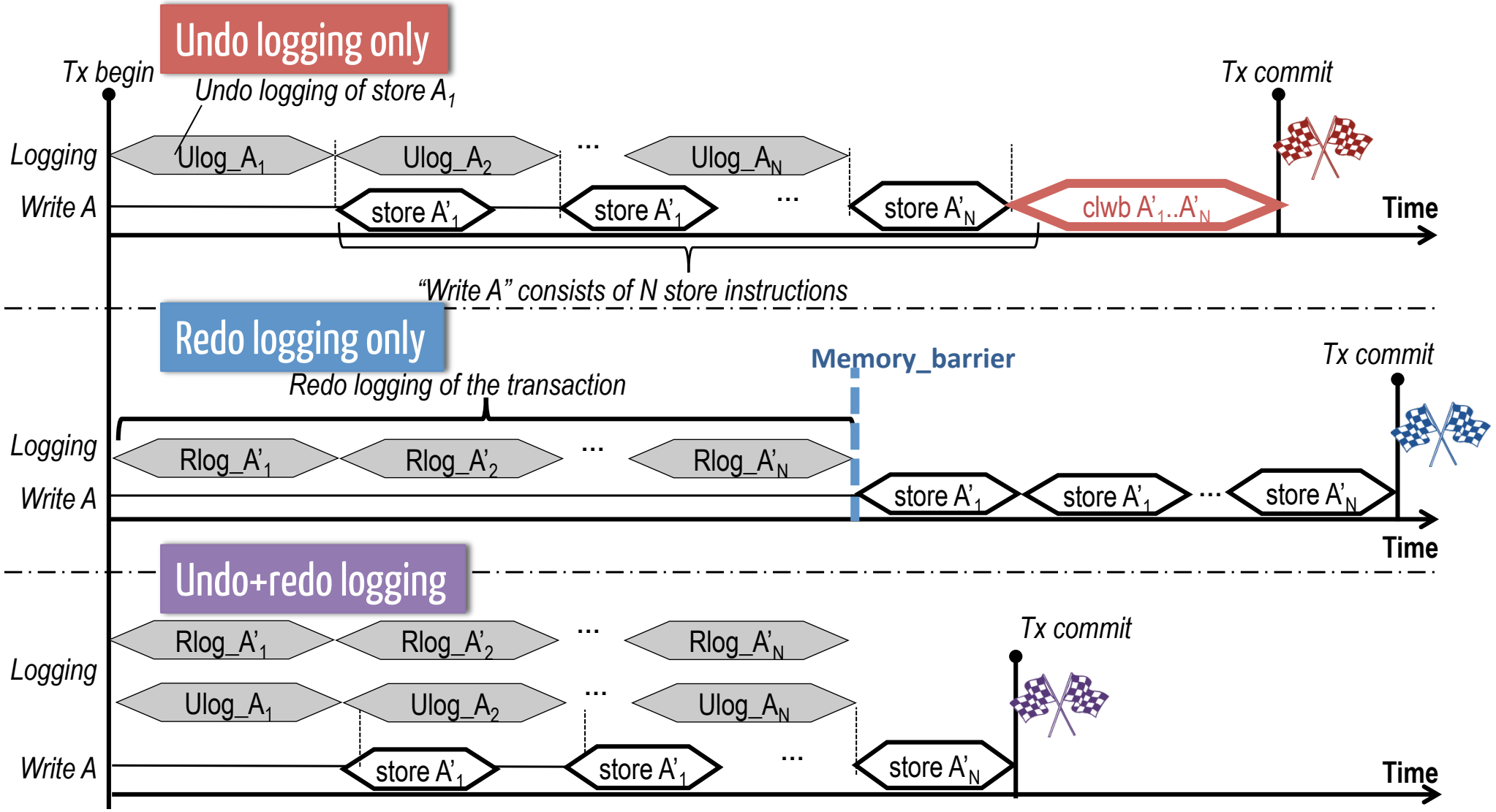
Benefits of undo + redo logging



Benefits of undo + redo logging



Preview of performance benefit with undo+redo logging



Inefficiency of logging by software in persistent memory

Increased memory traffic

Extra instructions in the CPU pipeline

Conservative cache flushes

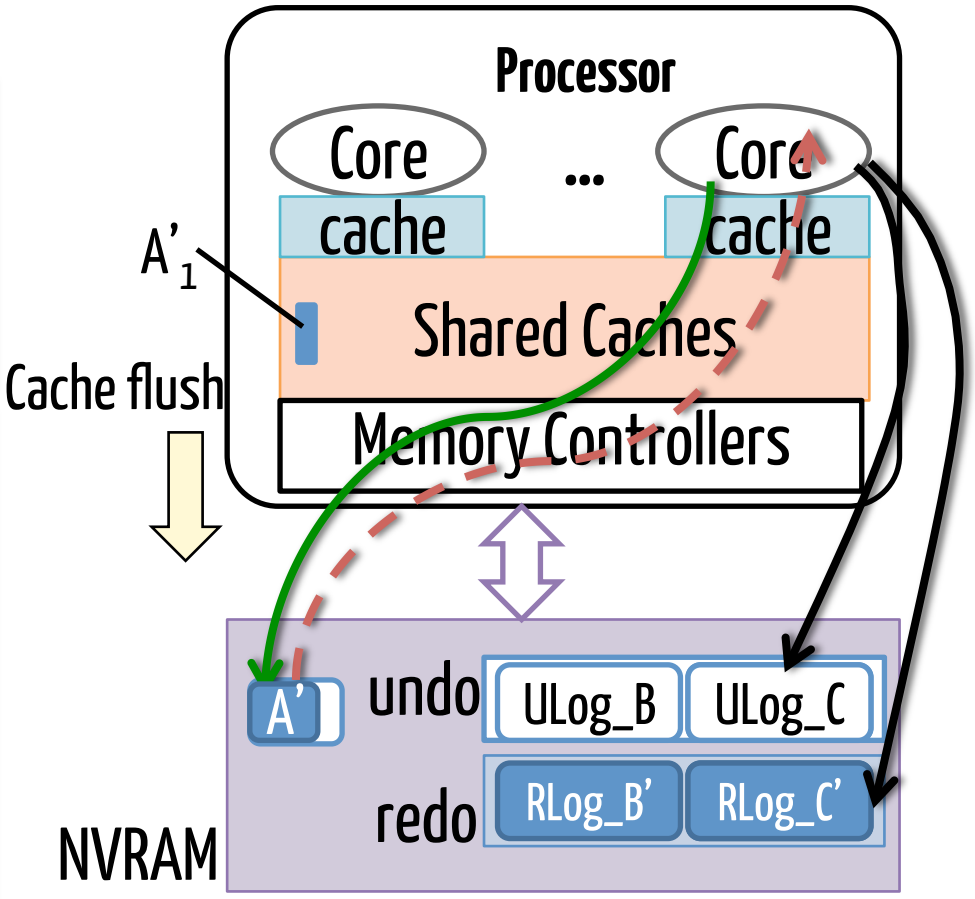
Risks with multithreading?

Transaction T_A:

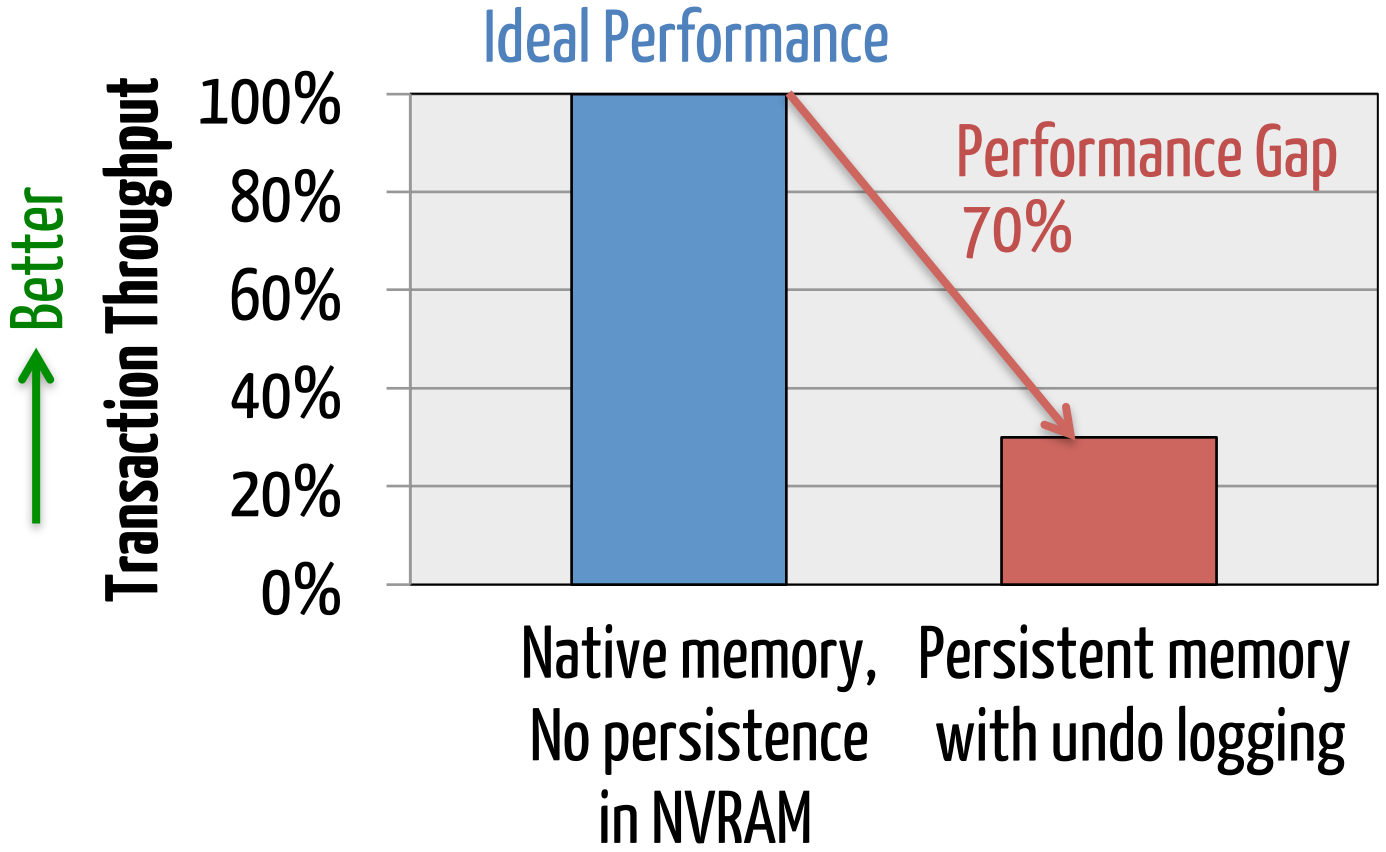
```
Tx_begin
   do some reads
   do some computation
   Uncacheable_log(addr(A),
                   new_val(A),
                   old_val(A))
   write A
   clwb    // can be delayed ?
Tx_commit
```

Micro-ops:
store log_A'$_1$
store log_A'$_2$
...

Micro-ops:
load A_1
load A_2
...
store log_A_1
store log_A_2
...



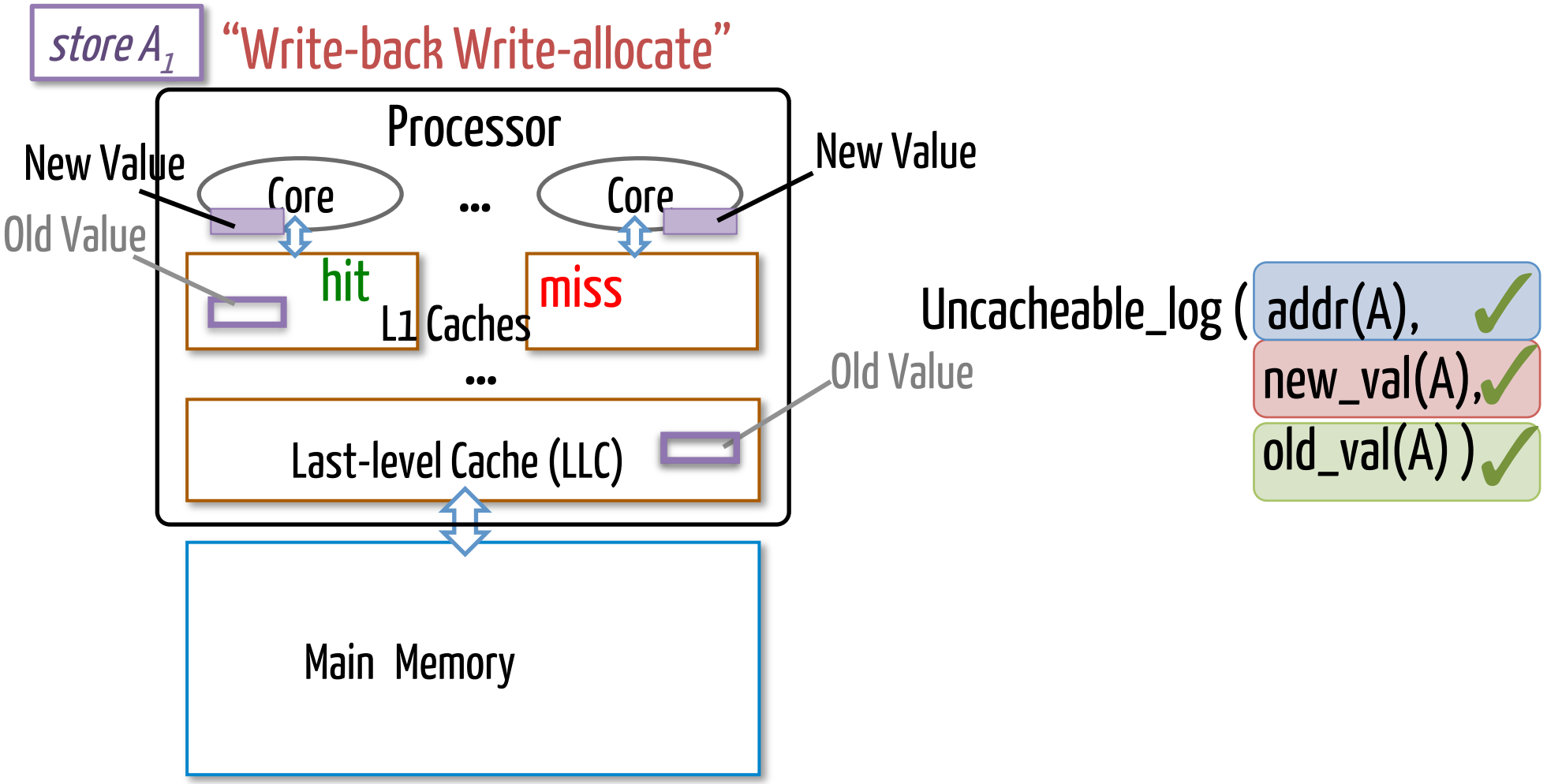
Performance cost of increased memory traffic



[Zhao +, MICRO'13]

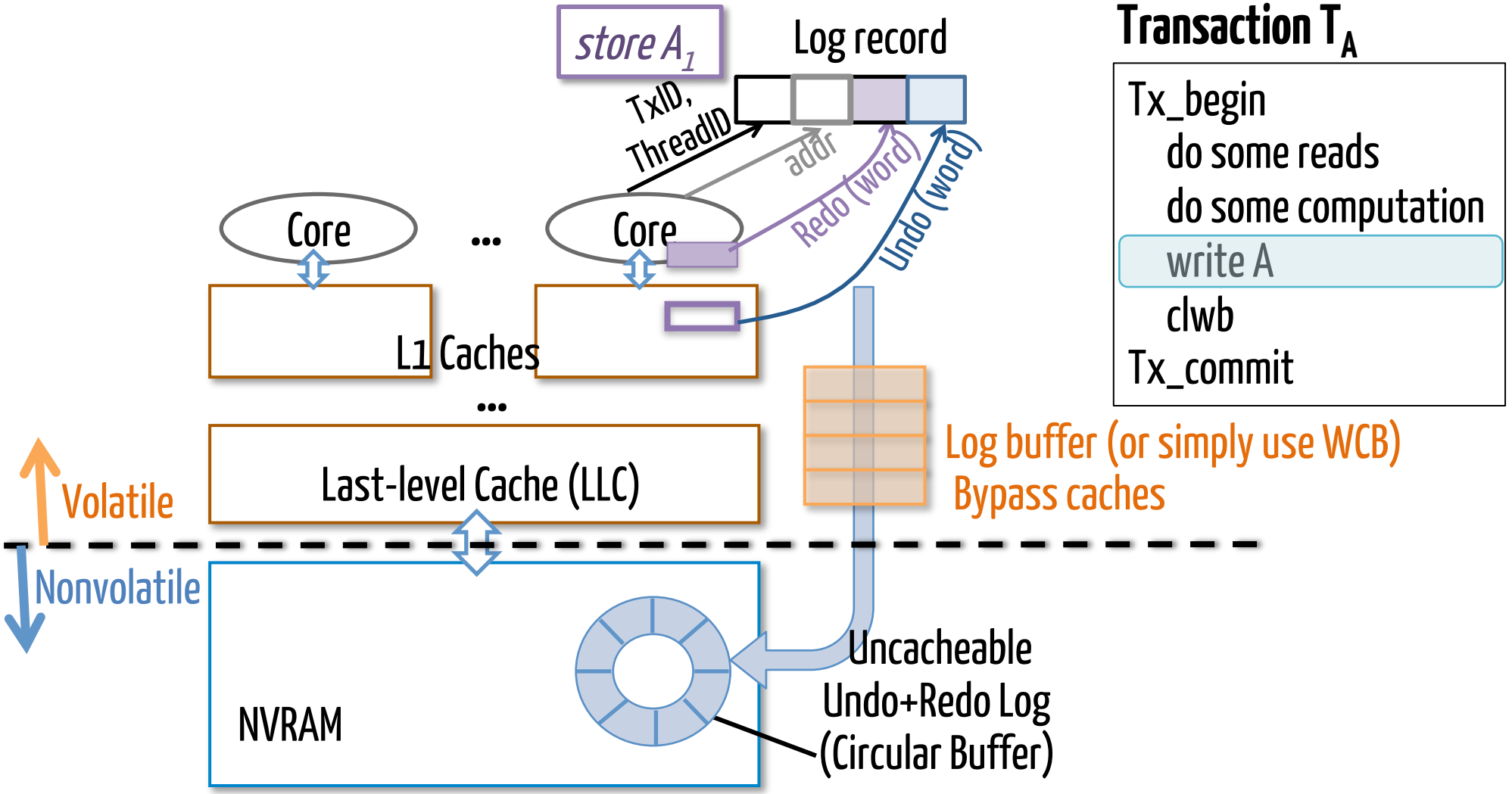
What can we leverage from the hardware?

Cache hierarchy maintains undo + redo information by nature



Our undo+redo logging: Take a ride given by CPU caching

Maintain the order between log and data updates by nature



How about cache flushes?

Decouple cache flushes and transaction execution

Existing in commodity processors

Multithreading?

fwb	dirty	data
1	1	
0	0	
1	1	
0	0	

CPU Caches

Volatile

Nonvolatile

The log is a circular buffer

Transaction T_A

```
Tx_begin
    do some reads
    do some computation
    write A
    clwb
Tx_commit
```

When is the appropriate time for cache flushes?

The need for cache flushes depends on log size and log update speed !

Trade-offs of hardware bookkeeping overhead, memory traffic bursts, and cache flush frequency

Commit the transaction

Design principles

- Undo+redo logging – taking a ride given by CPU caching
- Cache flushes – decoupled from transaction execution

Transaction T_A

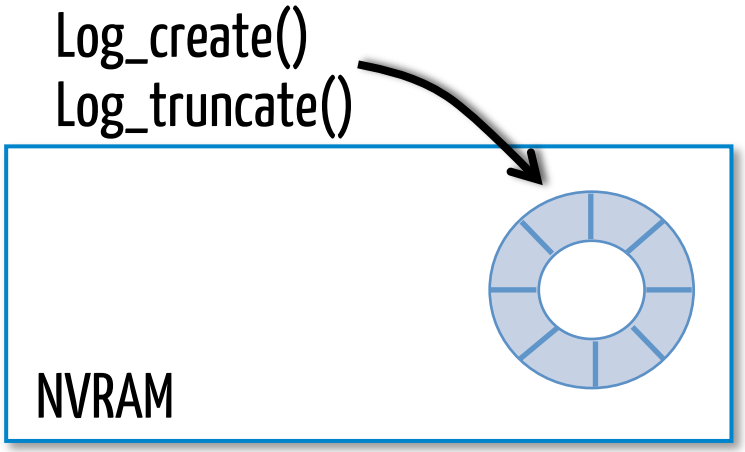
```
Tx_begin
    do some reads
    do some computation
    write A
    clwb
Tx_commit
```

Software and hardware implementation cost

- Software support

Transaction interface

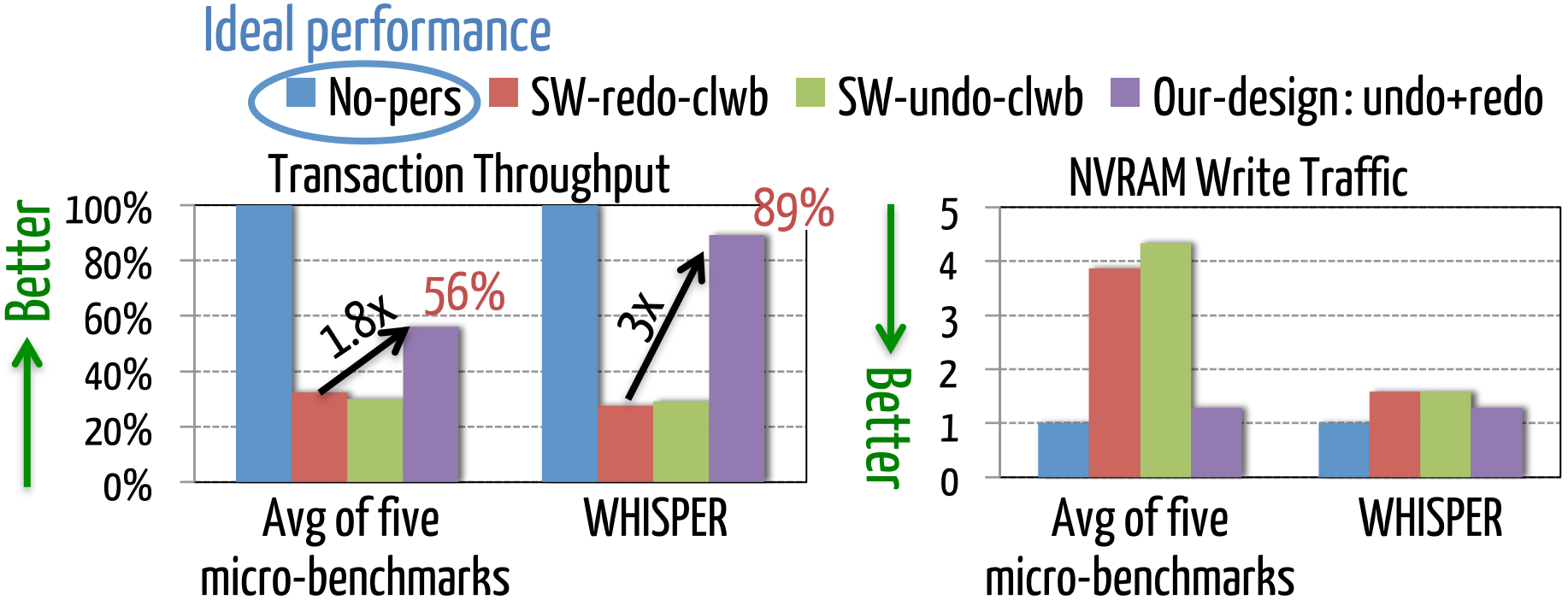
```
Tx_begin
    do some reads
    do some computation
    Write A
Tx_commit
```



- Hardware overhead

Major Components	Logic Type	Size
Transaction ID register	Flip-flop	1 byte per HW thread
Log head and tail registers	Flip-flop	16 bytes
Fwb cache tag bit	SRAM	1 bit per cache line

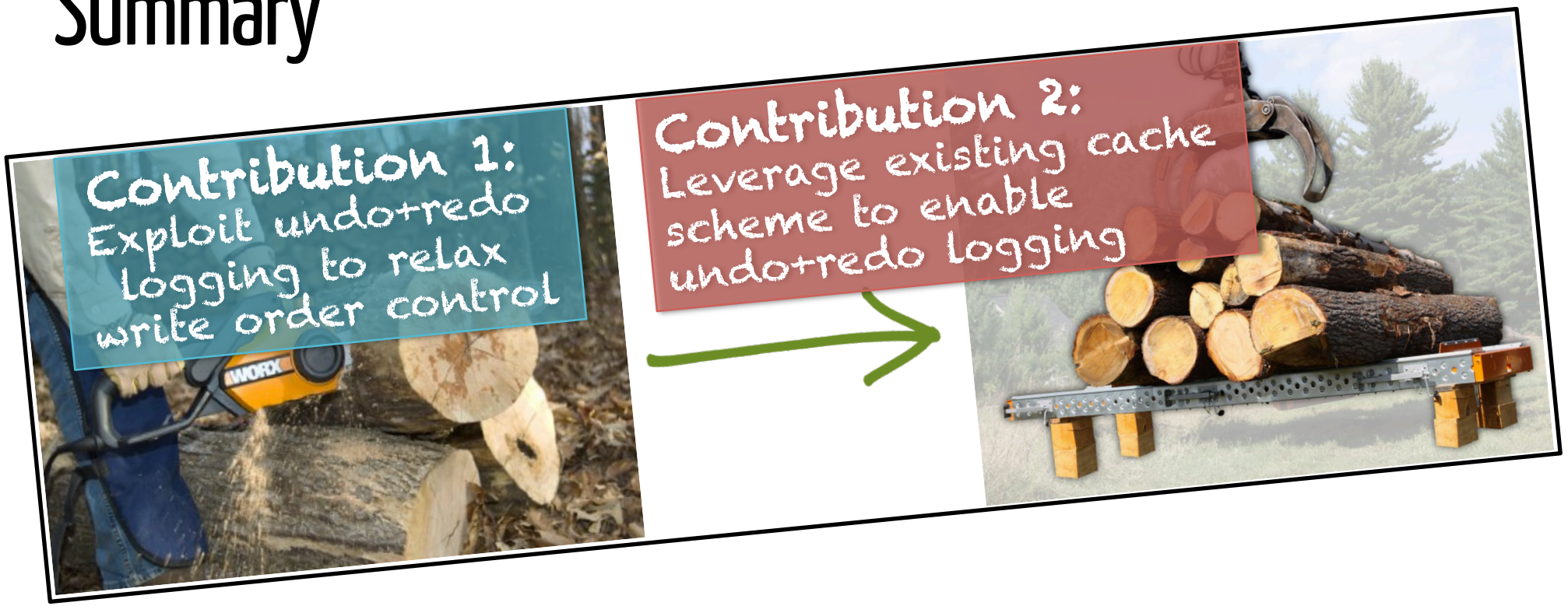
Key performance results



Processor configuration: Core i7, 22nm, 4-core, 2.5GHz, 2 threads/core

Other results: energy consumption, instruction increase, IPC, sensitivity studies, etc.

Summary



- Key points
  - Rethink the way traditional software schemes are used
  - Exploit opportunities in existing hardware – can support data persistence by nature

Hardware-driven Undo+Redo Logging

Jishen Zhao

https://users.soe.ucsc.edu/~jzhao/

Computer Engineering, UC Santa Cruz

October 11, 2017

HPTS
HIGH PERFORMANCE TRANSACTION SYSTEMS