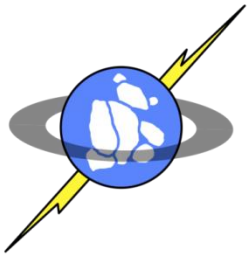
New Abstractions for Fast Non-Volatile Storage

Joel Coburn, Adrian Caulfield,

Laura Grupp, Ameen Akel, Steven Swanson

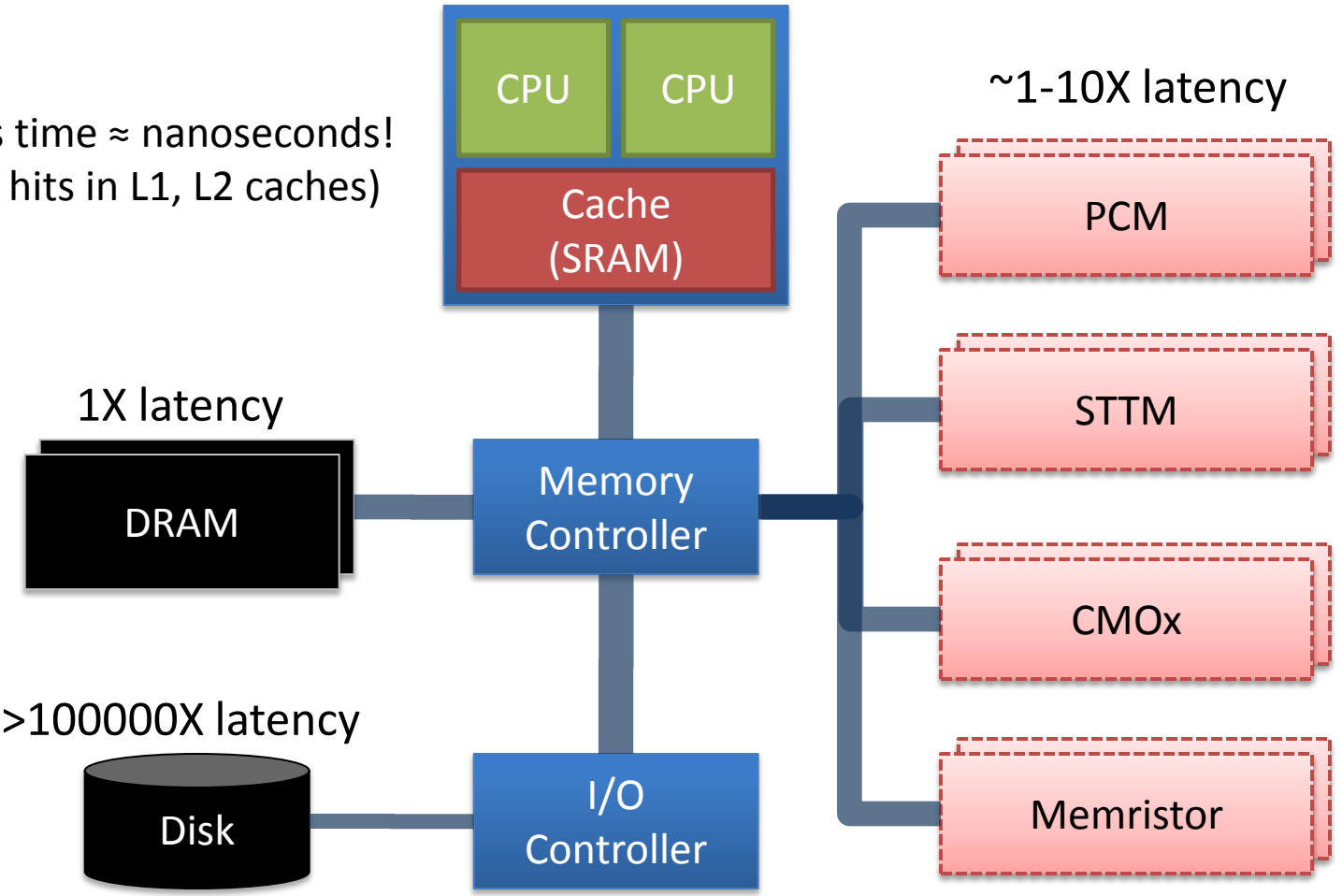
Non-volatile Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego



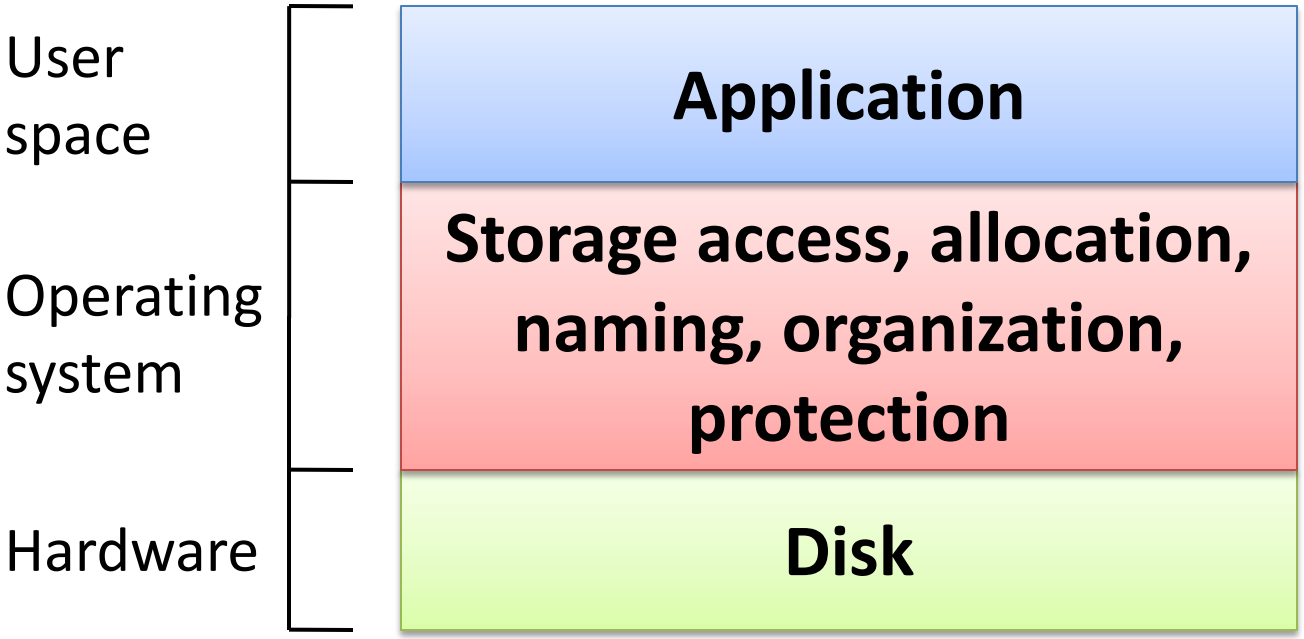
How do you re-engineer a system where disk is as fast as DRAM?

Storage will be as fast as DRAM

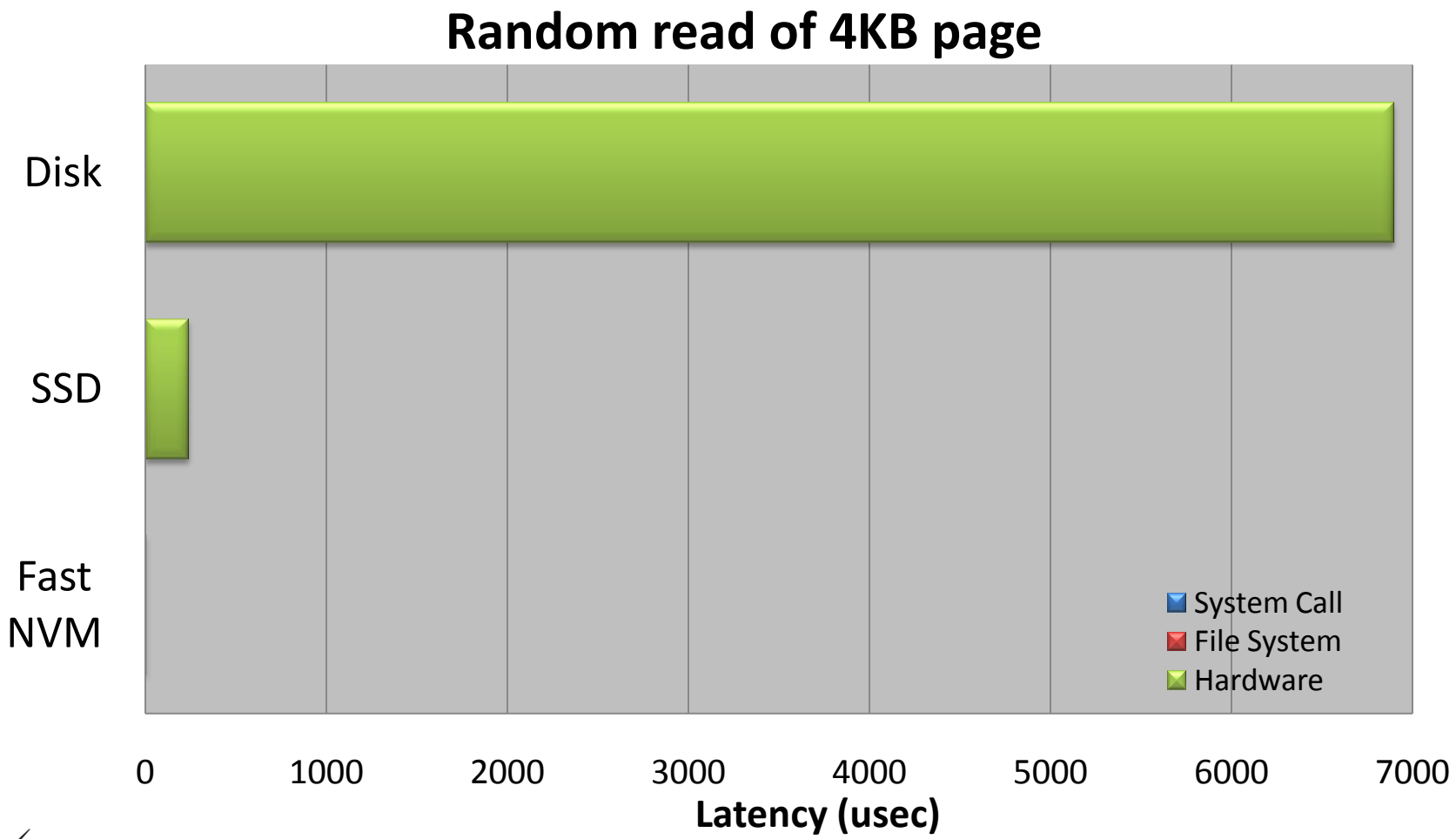
IO access time ≈ nanoseconds!
(if a read hits in L1, L2 caches)



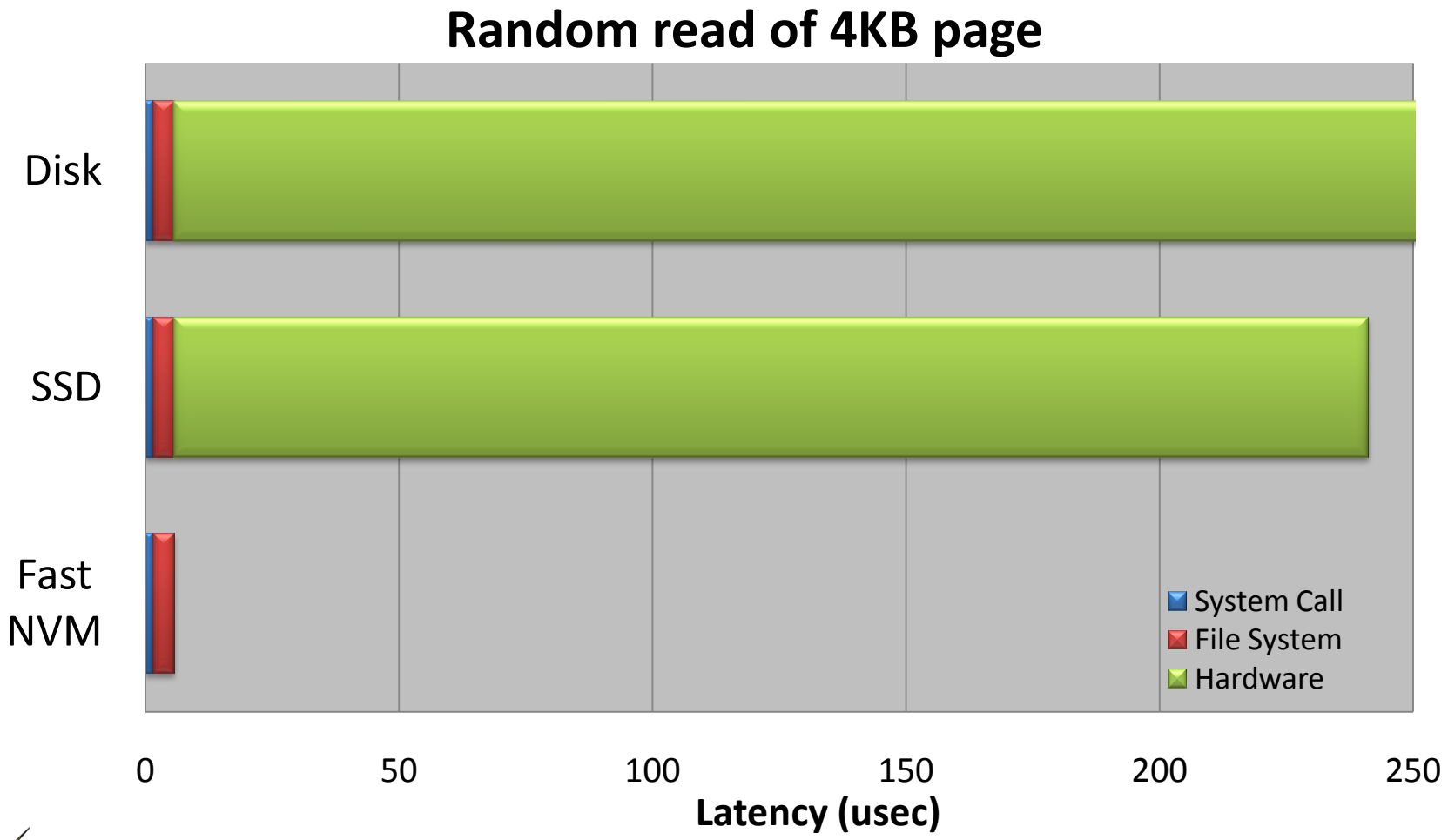
System Stack for Disk



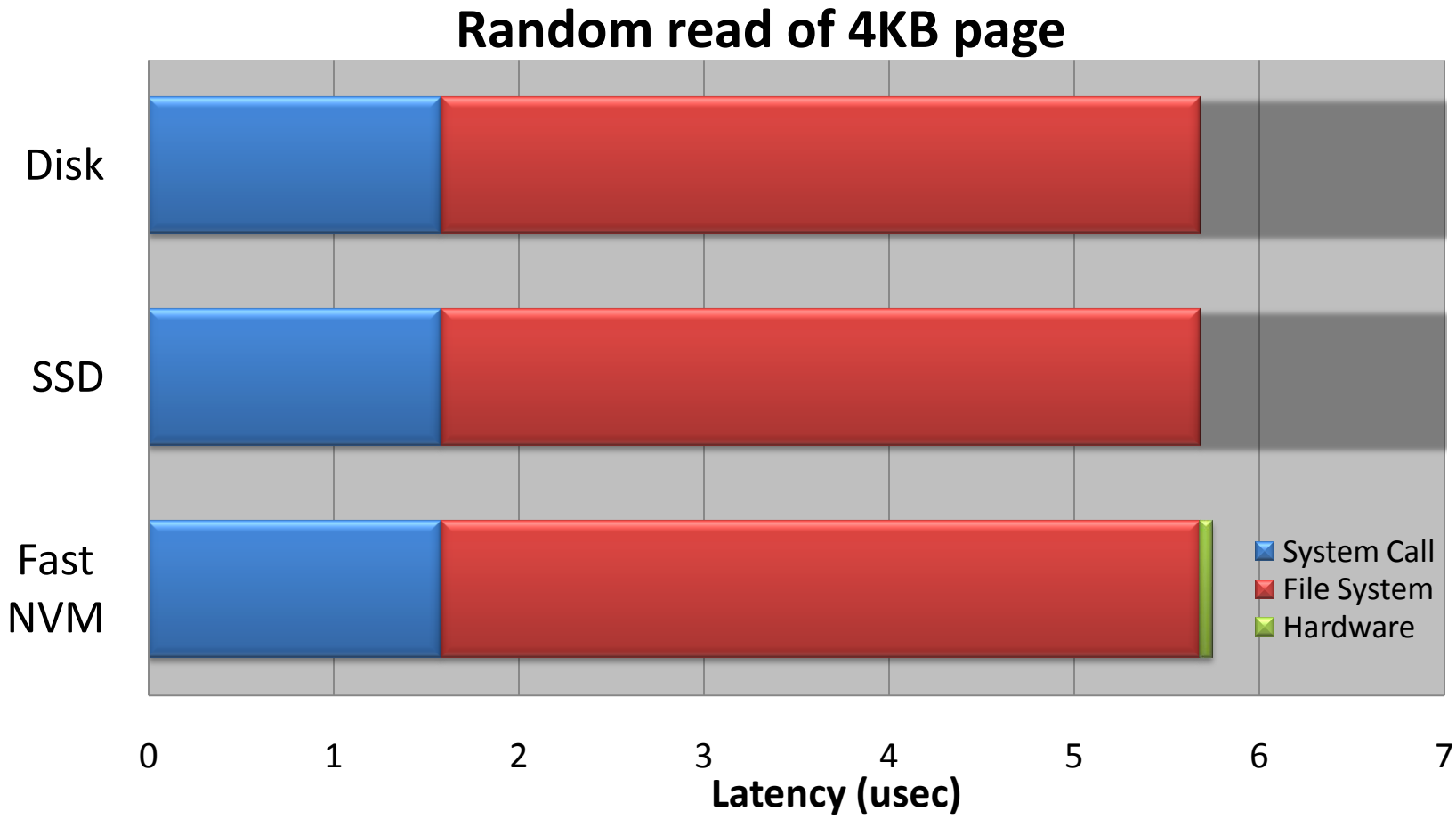
Yesterday's Interfaces for Tomorrow's Storage



A closer look…



Amdahl's Law strikes again!

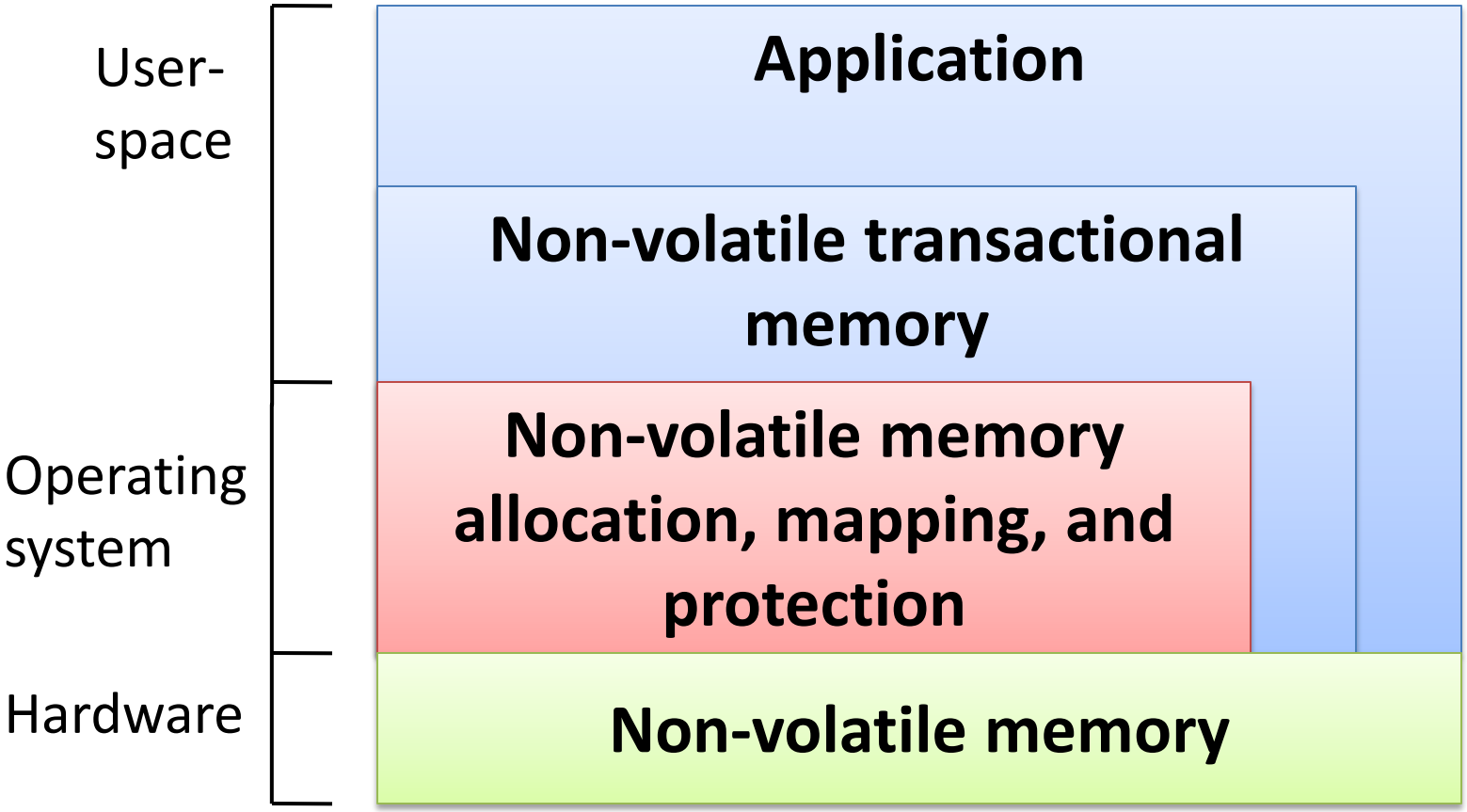


The existing system stack makes fast NVMs 100X slower.

Criteria for New Abstractions

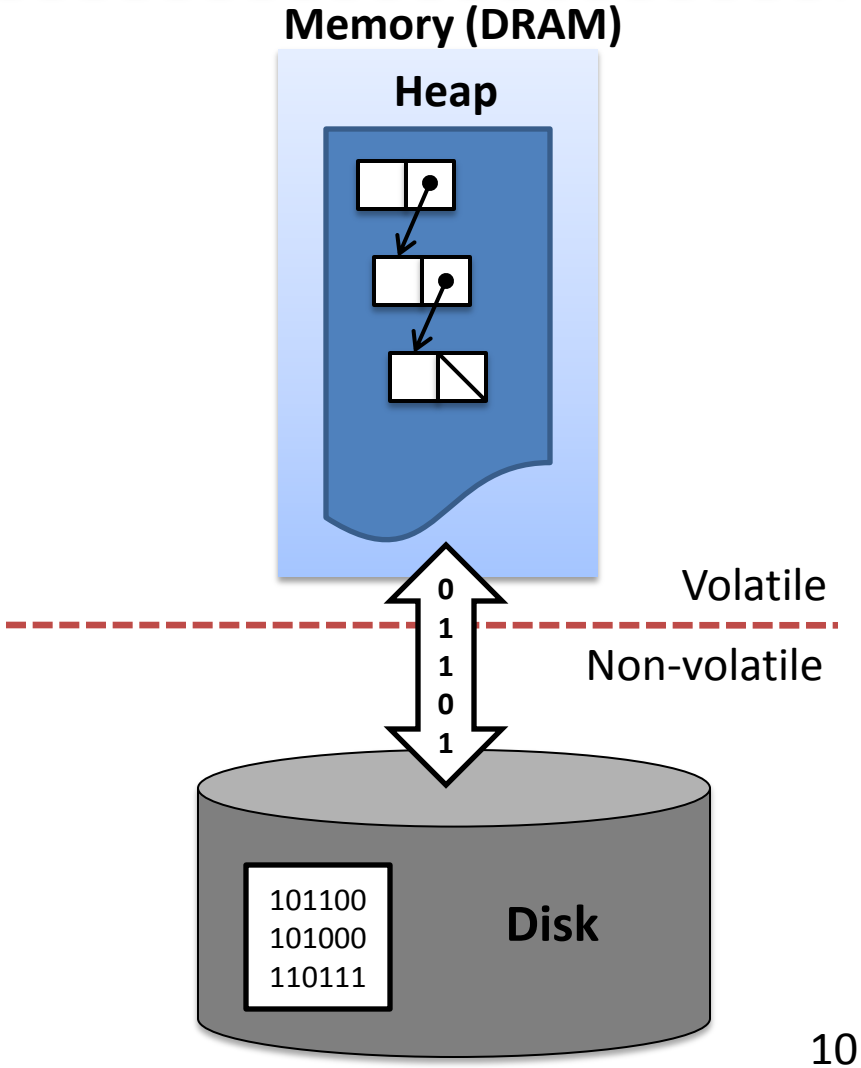
- ***Fast***
  - Application level latency is as close as possible to the latency of the storage technology
  - Memory-like interface—direct load/store access to non-volatile storage
- ***Persistence & reliability***
  - Data structures should be robust against failures
- ***Flexibility***
  - Few, if any, restrictions on how data is accessed
- ***Organization, sharing, protection***

Non-Volatile Transactional Memory



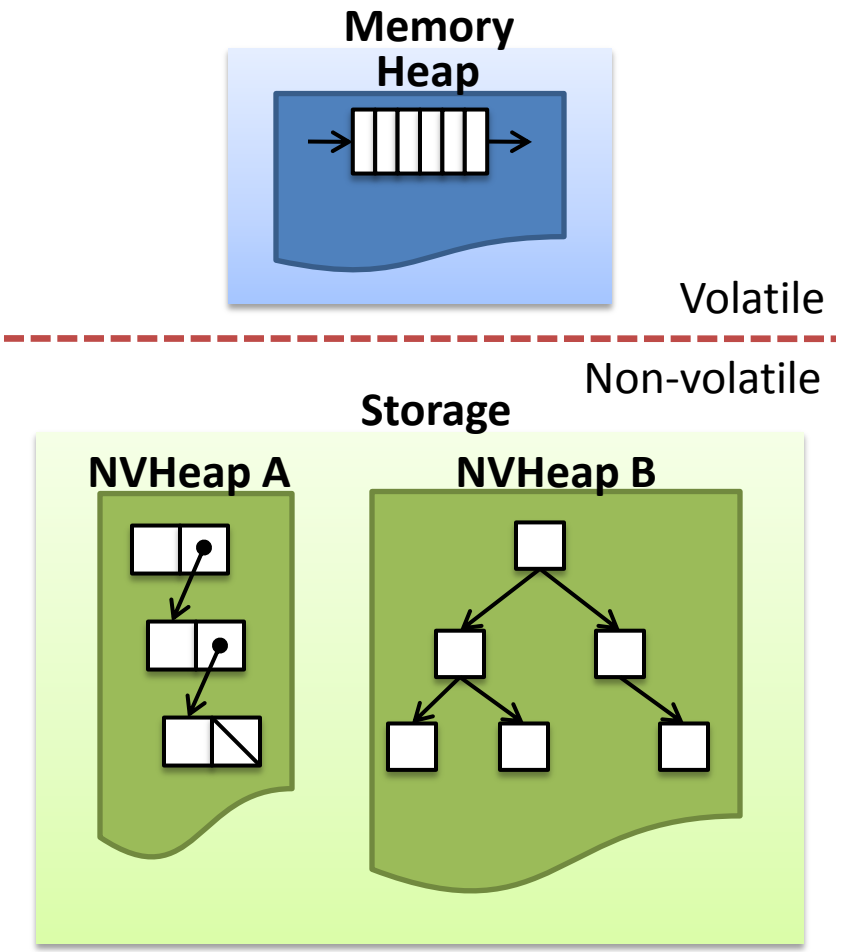
The Old Way: Volatile Heap

- A program accesses a single heap in volatile memory
  - Can build rich, pointer-based data structures
- Serialize/de-serialize to transfer data to/from storage



The New Way: Non-Volatile Heaps

- A program can have multiple heaps in non-volatile storage
  - Appear in the application's address space just like memory
  - Rich, pointer-based data structures live directly in storage
  - Provides safety guarantees



Accessing Non-Volatile Heaps: Transactions

- Definition: a sequence of operations that is atomic and durable
- What transactions buy us
  - Failure recovery
  - Concurrency management
- Used by databases to make guarantees about storage

Transactions: How they work

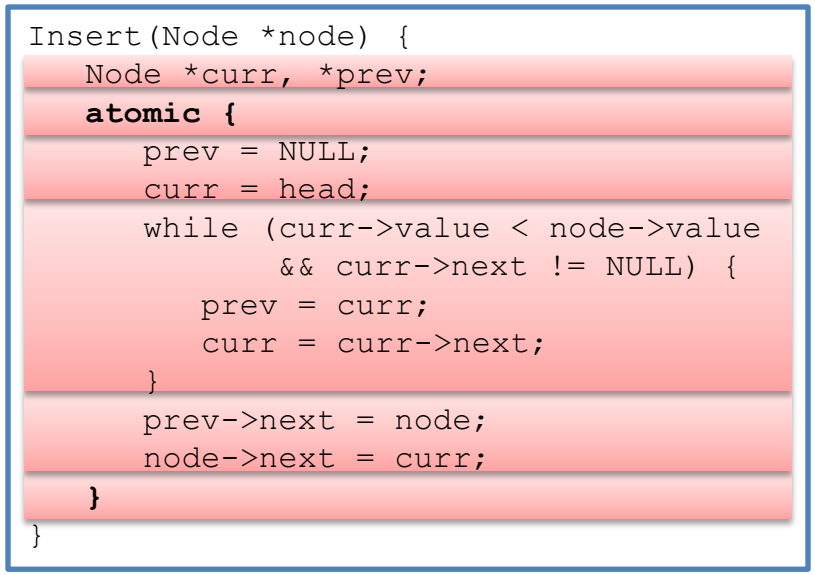
1. Perform operations
   - Record the state of data that is read or written in a log
2. Check for conflicts
   - Occur when data is changed outside of the transaction
3. If no conflicts, then commit (makes permanent)
4. If conflicts, then abort and rollback (undo)
   - Replay the logs to restore old values

A Simple Example

- We have a linked list in non-volatile storage
- We want to insert a node in sorted order
- Requirement: If we crash, our list must remain in a good state
  - Either the node is added or it is not
  - No wild pointers, missing links, etc.

A Simple Example

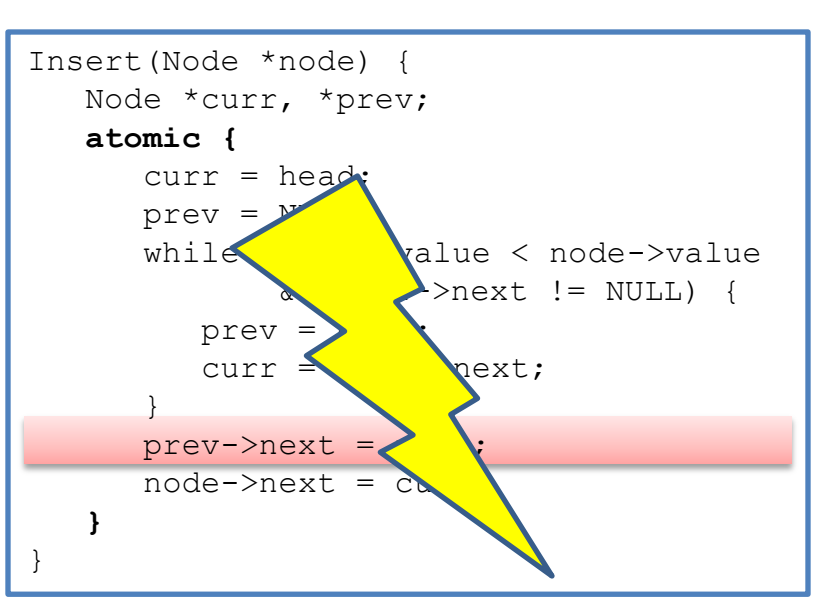
```
Insert(Node *node) {
   Node *curr, *prev;
   atomic {
      prev = NULL;
      curr = head;
      while (curr->value < node->value
             && curr->next != NULL) {
         prev = curr;
         curr = curr->next;
      }
      prev->next = node;
      node->next = curr;
   }
}
```

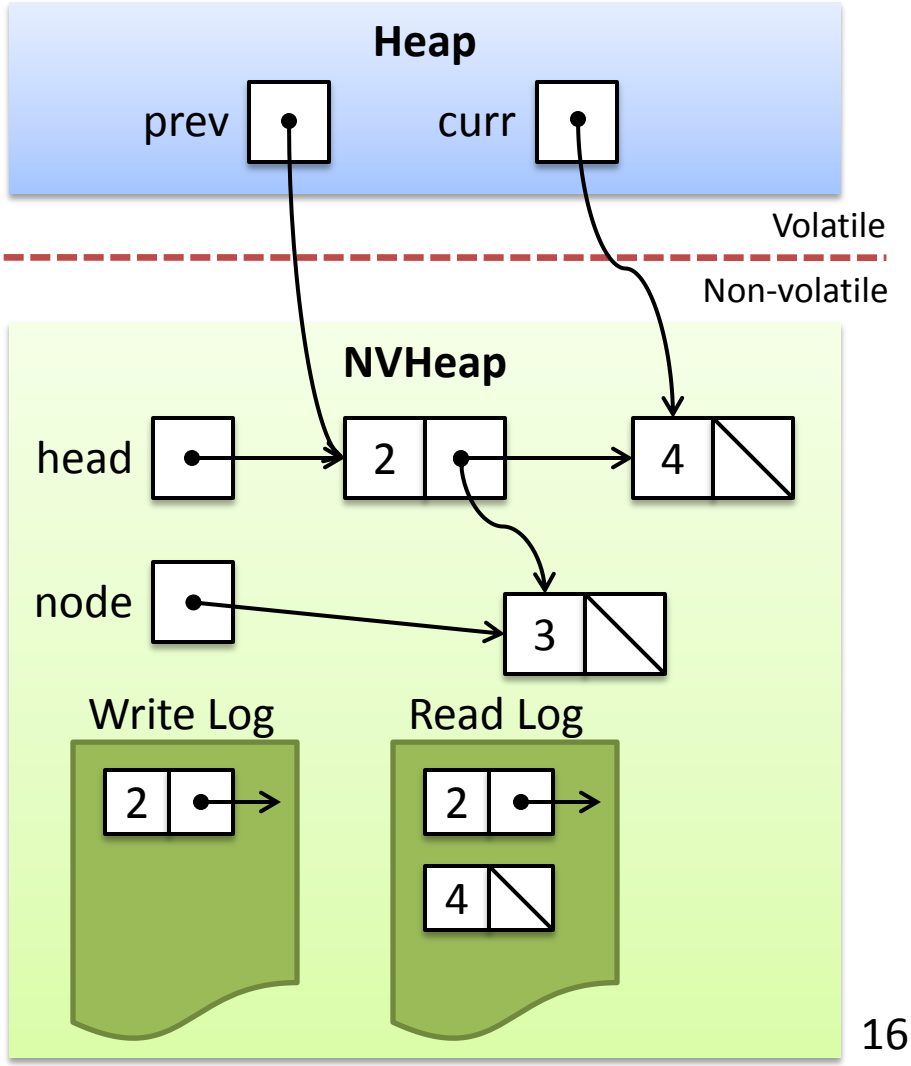


Transaction Commit

A Simple Example

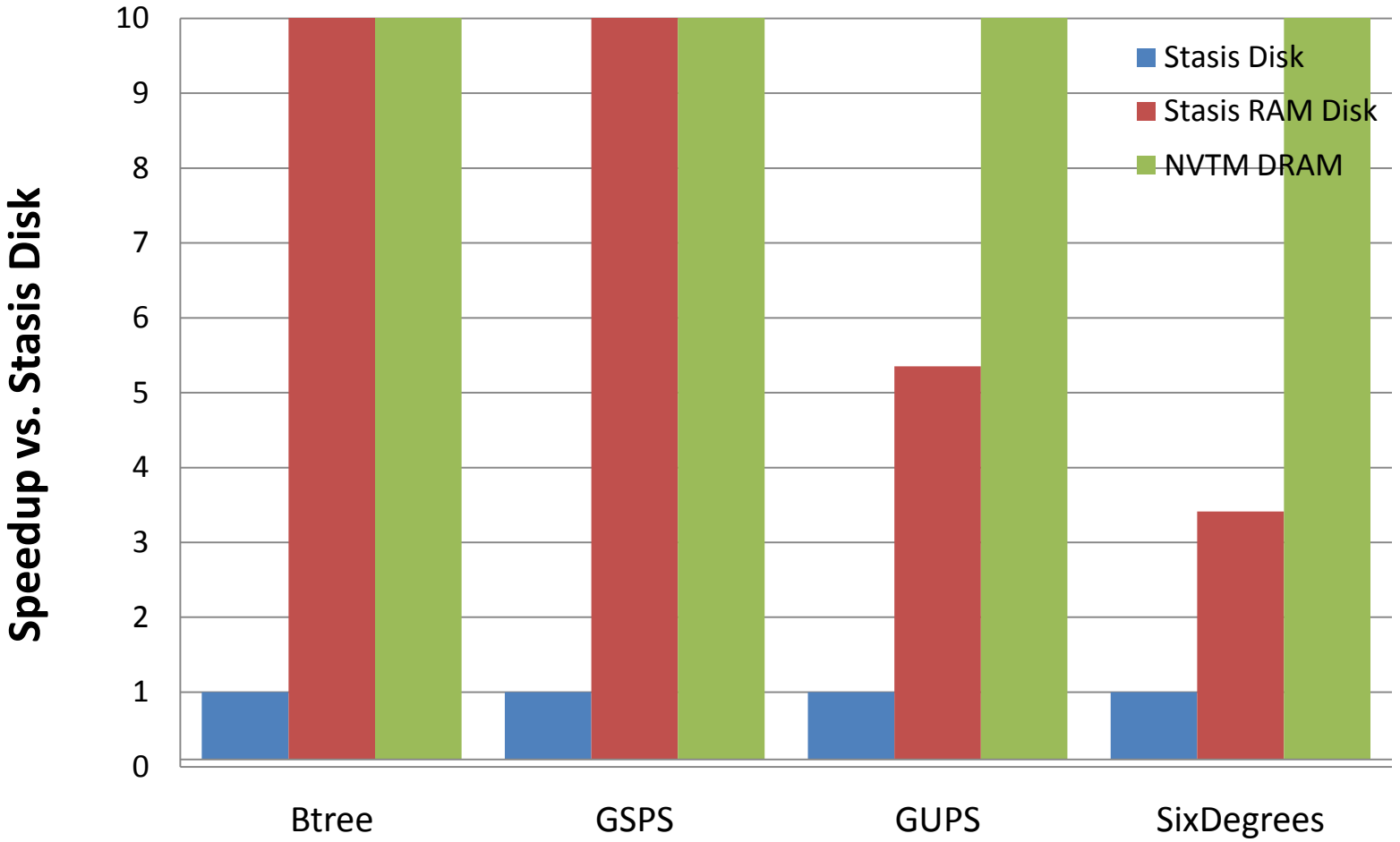
```
Insert(Node *node) {
   Node *curr, *prev;
   atomic {
      curr = head;
      prev = N
      while          value < node->value
                &      ->next != NULL) {
         prev =
         curr =       next;
      }
      prev->next =      ;
      node->next = cu
   }
}
```



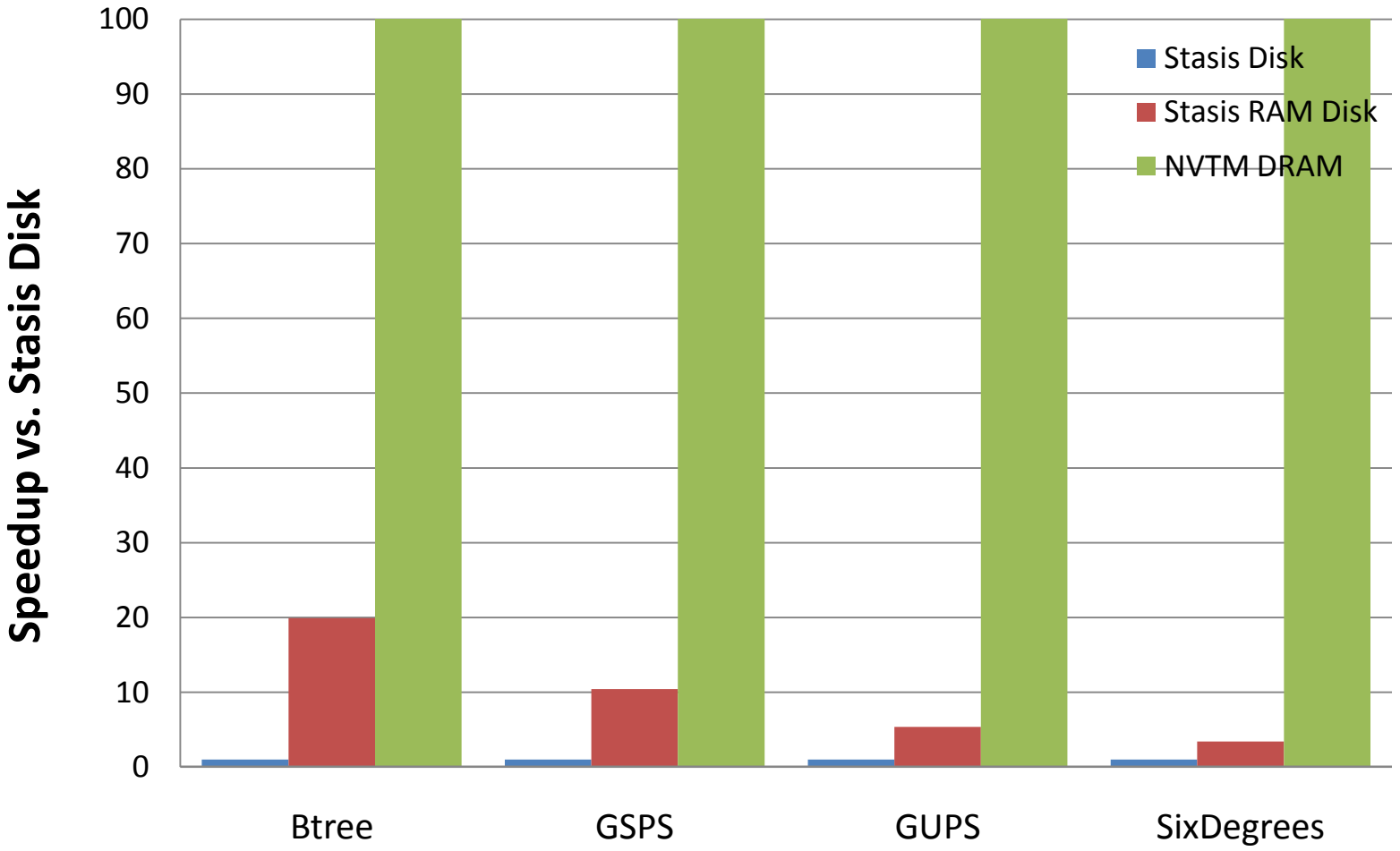


Transaction Abort

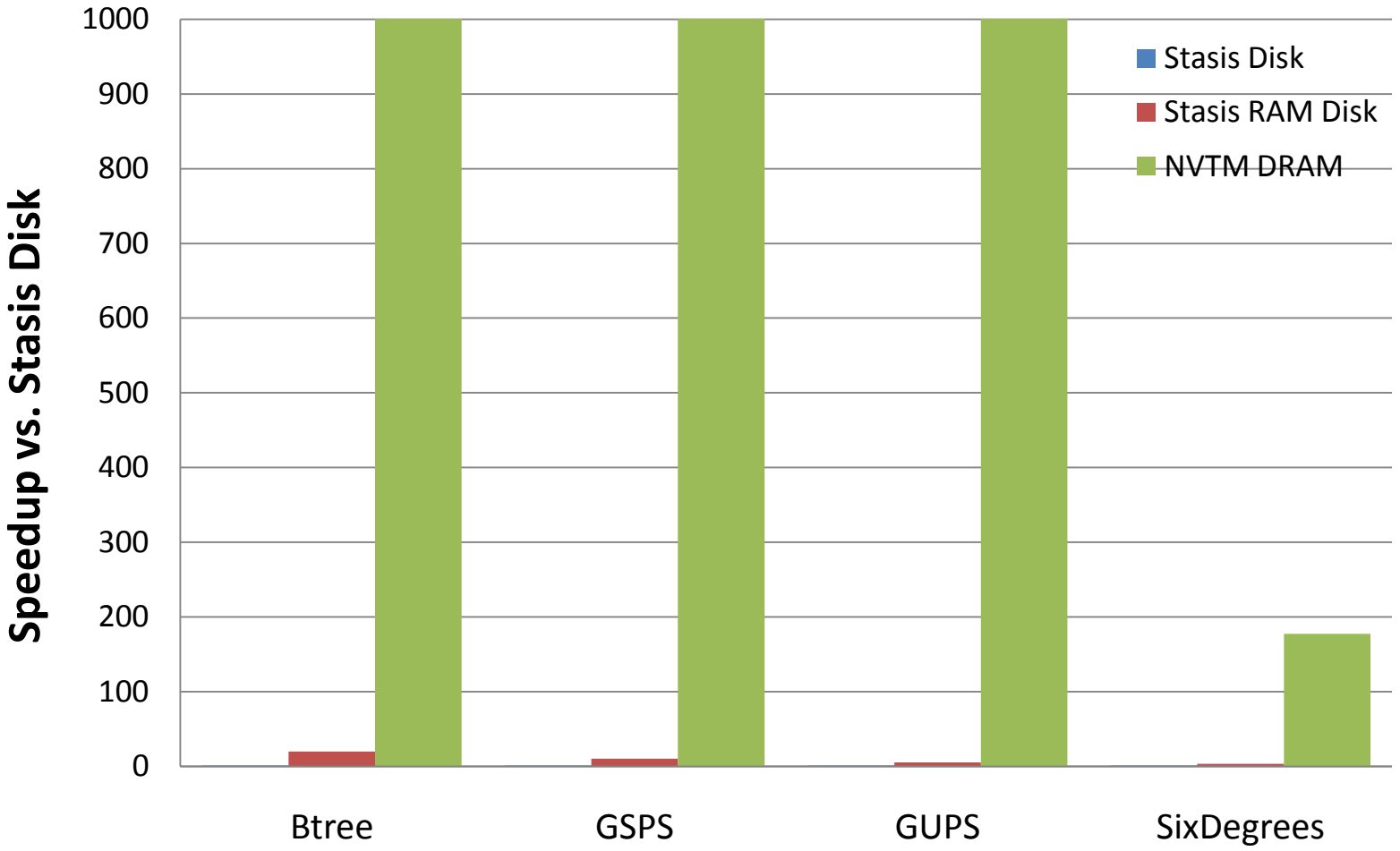
Results


Speedup vs. Stasis Disk
10
9
8
7
6
5
4
3
2
1
0
Stasis Disk
Stasis RAM Disk
NVTM DRAM
Btree
GSPS
GUPS
SixDegrees

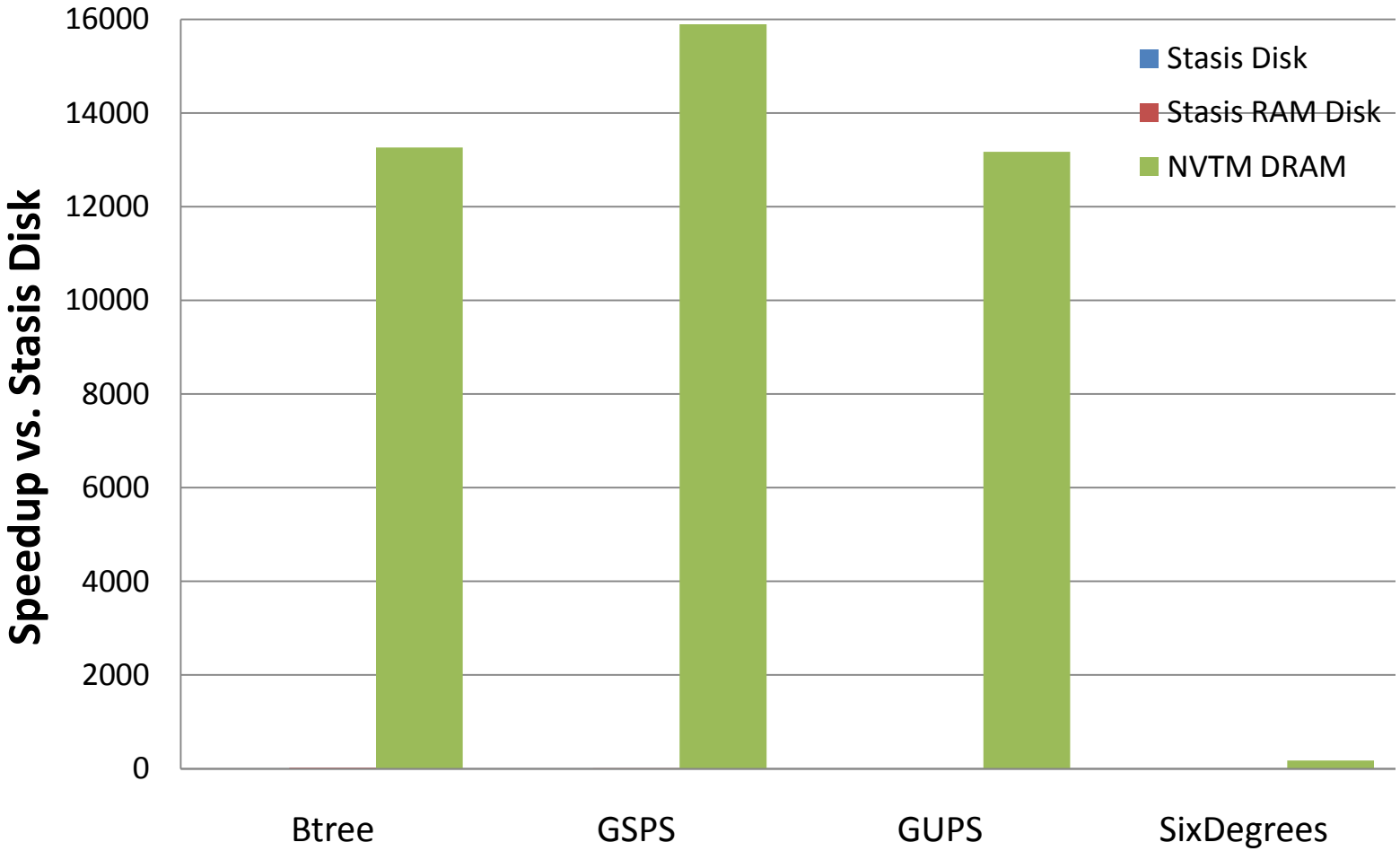
Results


Speedup vs. Stasis Disk
100
90
80
70
60
50
40
30
20
10
0
Stasis Disk
Stasis RAM Disk
NVTM DRAM
Btree
GSPS
GUPS
SixDegrees

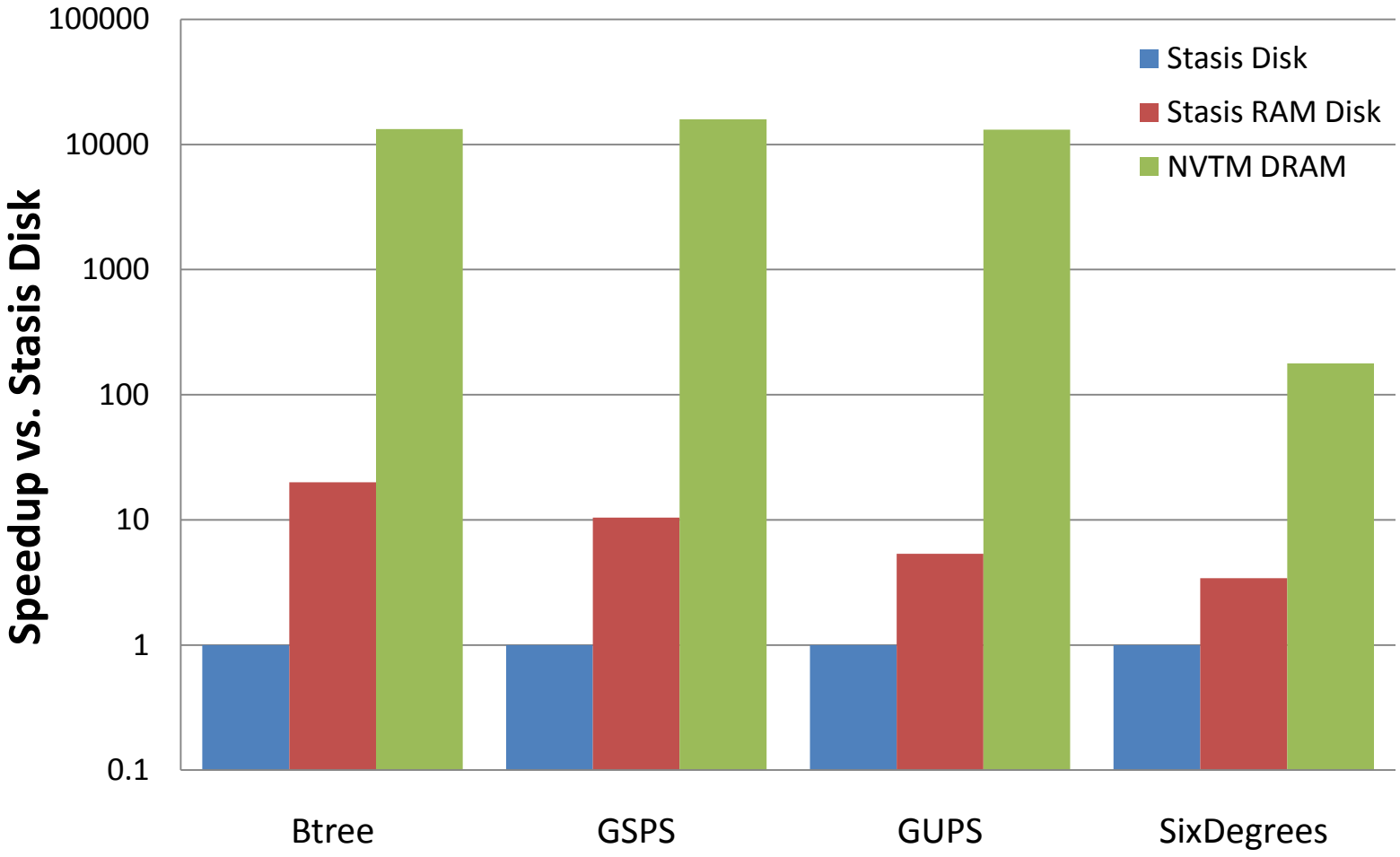
Results


Speedup vs. Stasis Disk
1000
900
800
700
600
500
400
300
200
100
0
Stasis Disk
Stasis RAM Disk
NVTM DRAM
Btree
GSPS
GUPS
SixDegrees

Results


16000
14000
12000
10000
8000
6000
4000
2000
0
Speedup vs. Stasis Disk
Stasis Disk
Stasis RAM Disk
NVTM DRAM
Btree
GSPS
GUPS
SixDegrees

Results


Speedup vs. Stasis Disk
100000
10000
1000
100
10
1
0.1
Stasis Disk
Stasis RAM Disk
NVTM DRAM
Btree
GSPS
GUPS
SixDegrees

Conclusion

- New non-volatile memory technologies are changing the way we think about storage
  - DRAM-like performance (both sequential and random access) + density + persistence
- Re-designing the storage abstraction will realize the full potential of these technologies!
  - Non-volatile transactional memory

Thank you!
Questions?