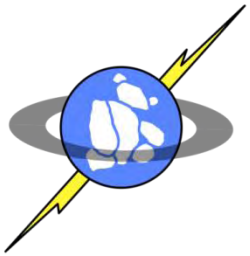
New Abstractions for Fast Non-Volatile Storage

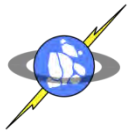
Joel Coburn, Adrian Caulfield,

Laura Grupp, Ameen Akel, Steven Swanson

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

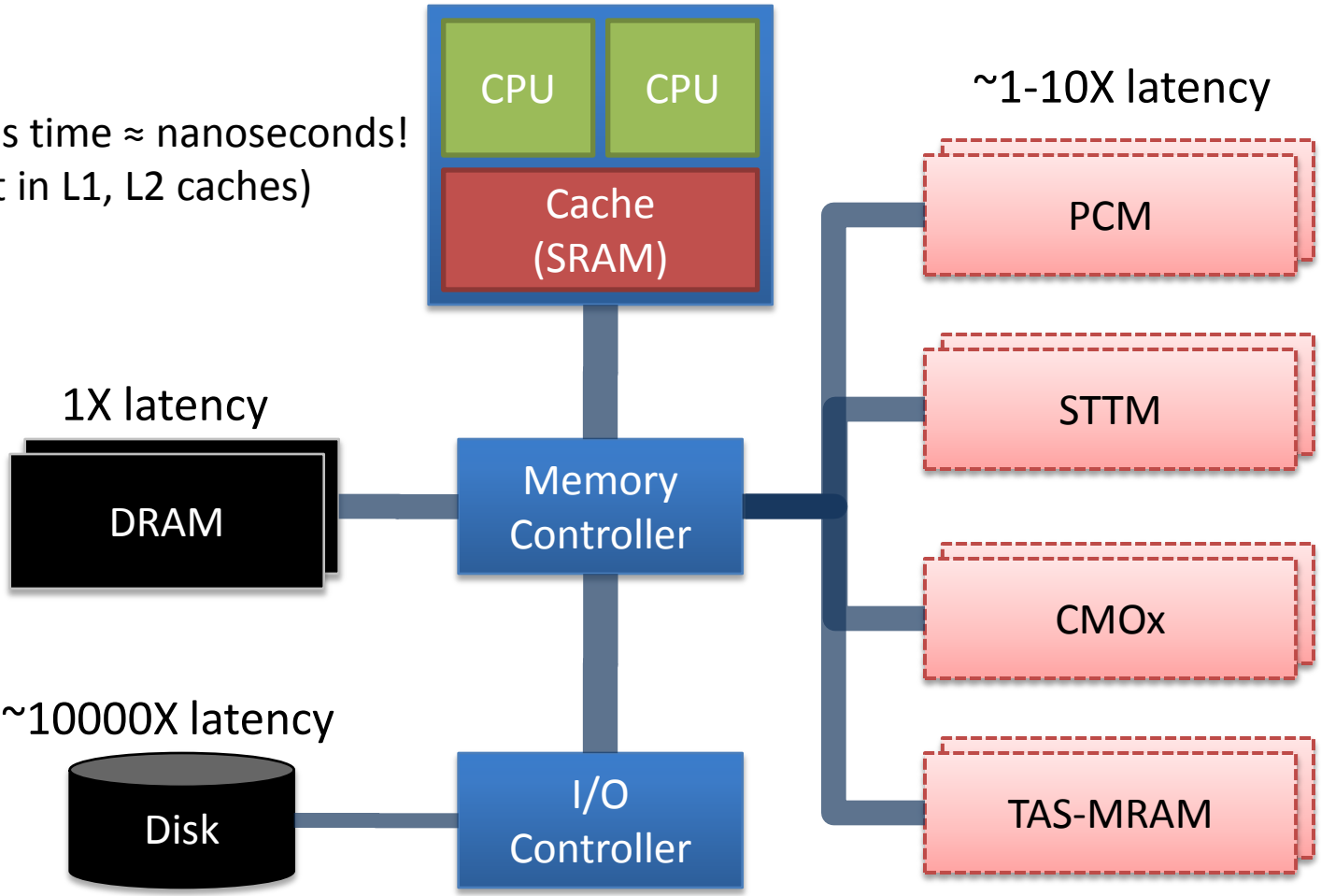


How do you re-engineer a system where disk is as fast as DRAM?

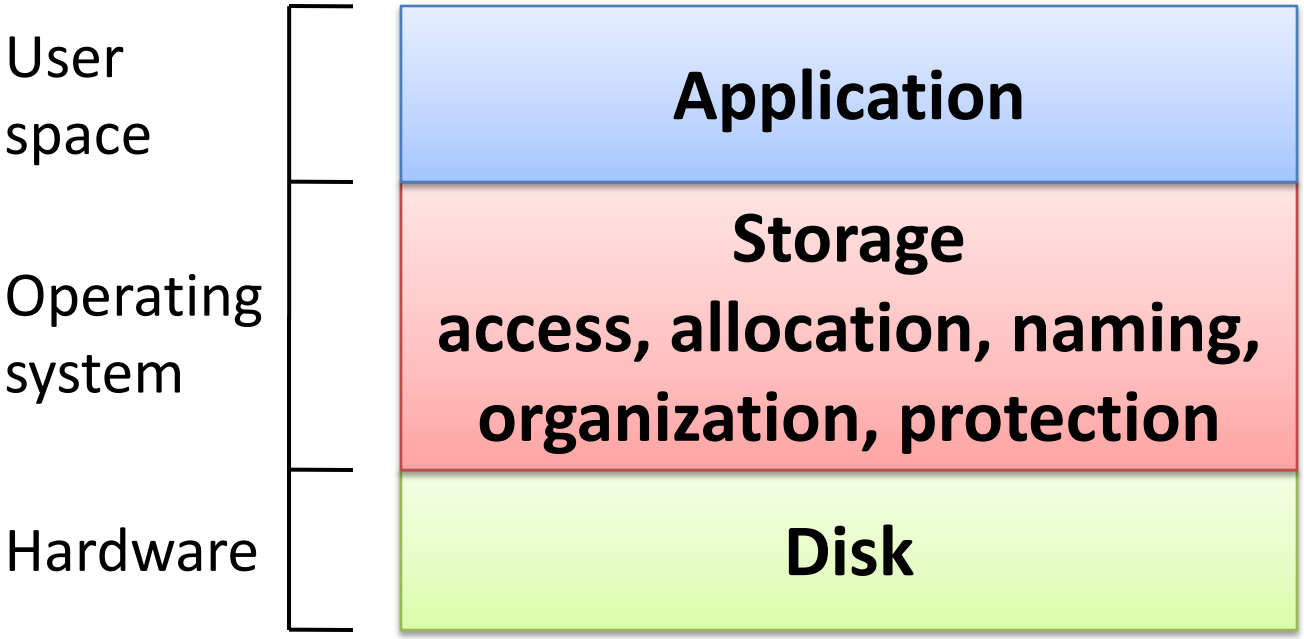


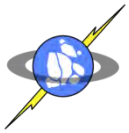
Storage System Architecture of the Future

IO access time ≈ nanoseconds!
(read hit in L1, L2 caches)

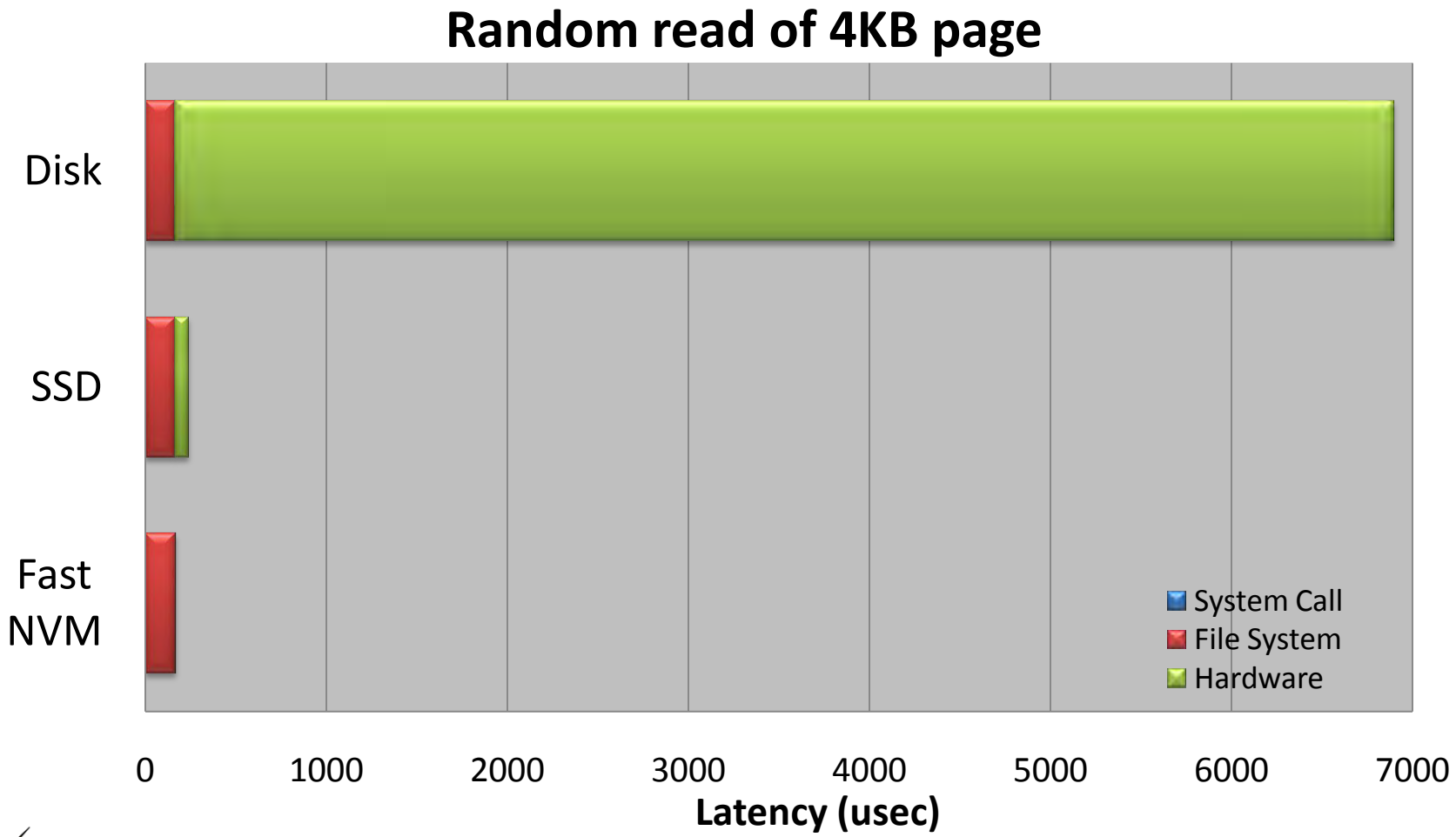


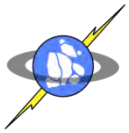
System Stack for Disk



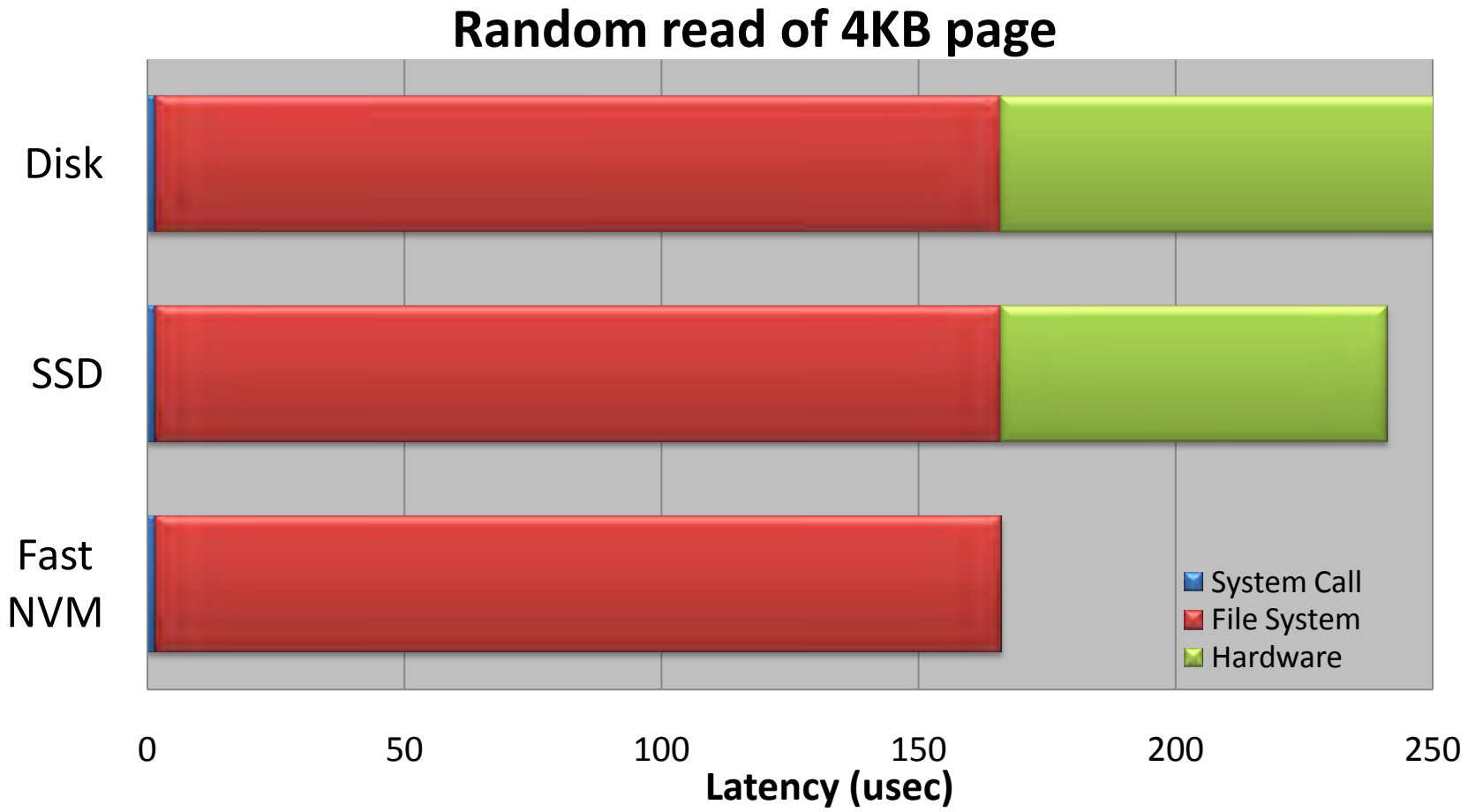


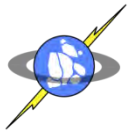
Yesterday's Interfaces for Tomorrow's Storage





Amdahl's Law strikes again!





Why do we have file systems?

- Access data
- Persistence & consistency (safety)
  - Holds data until it is explicitly deleted
  - Recovers from system crashes

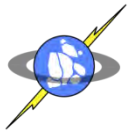
Needs to be fast

- Other stuff
  - Structure/organization for data (directories)
  - Allows sharing of data between processes, users, and machines
  - Provides protection from data theft or destruction (security)
  - Basic operations: create, delete, open

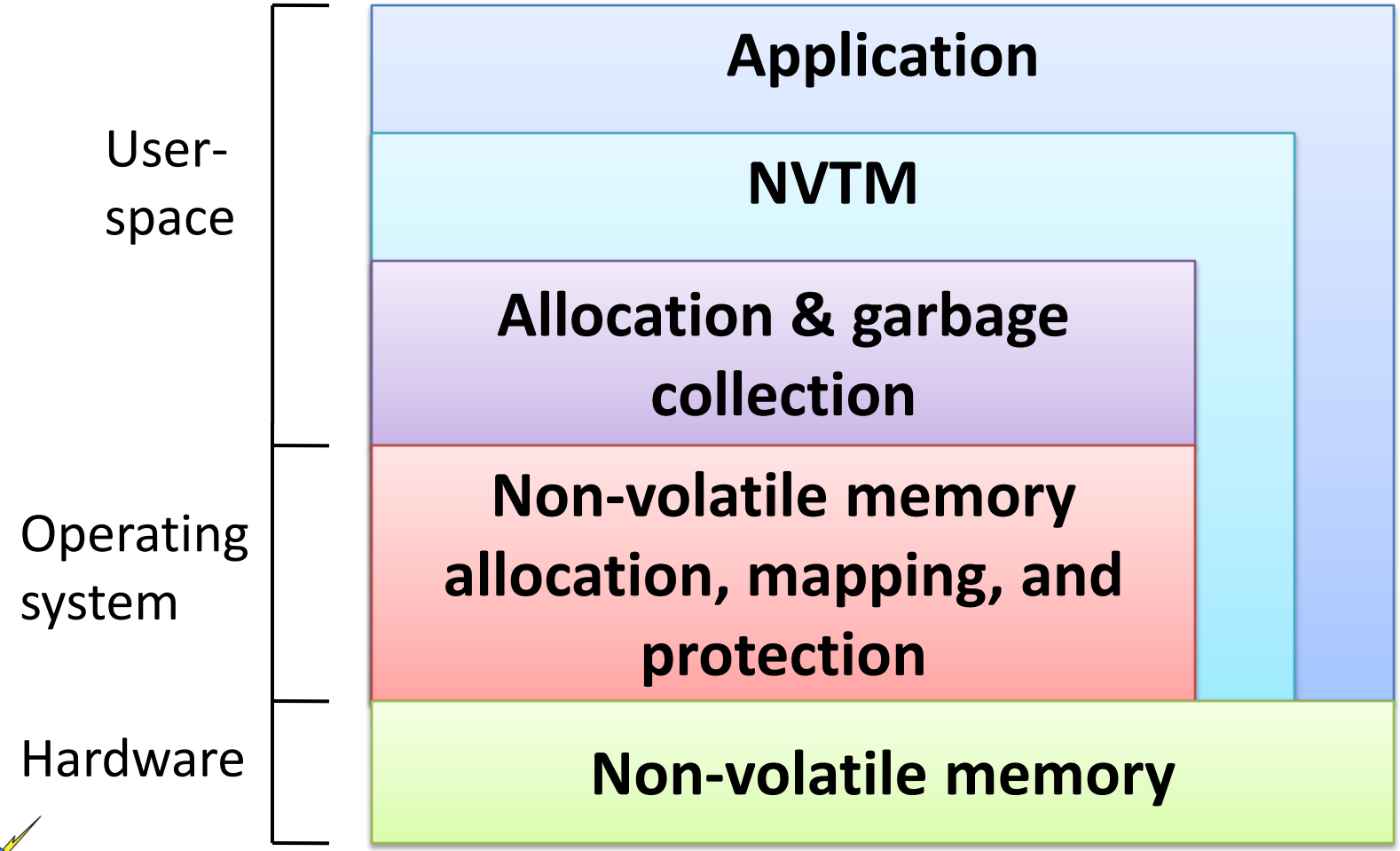
Can be slow

Criteria for New Abstractions

- Memory-like interface
  - Rich, non-volatile data structures implemented in the same way that volatile data structures are implemented—direct load/store access to memory
- Persistence and safety
  - Data structures should be robust against application and system failure
- High performance
  - Application level access latency should as close as possible to latency of the storage technology
  - Leverage caching in the memory hierarchy

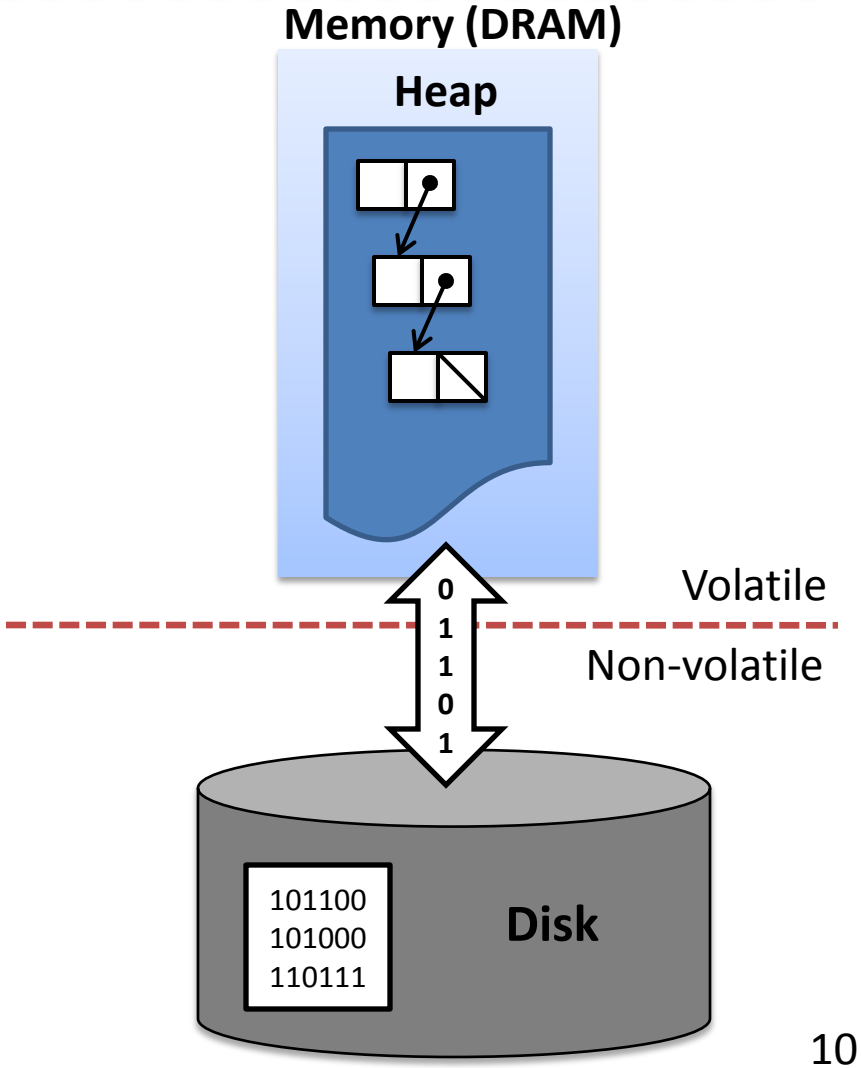


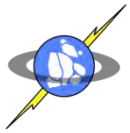
System Stack for Non-Volatile Memories



The Old Way: Volatile Heap

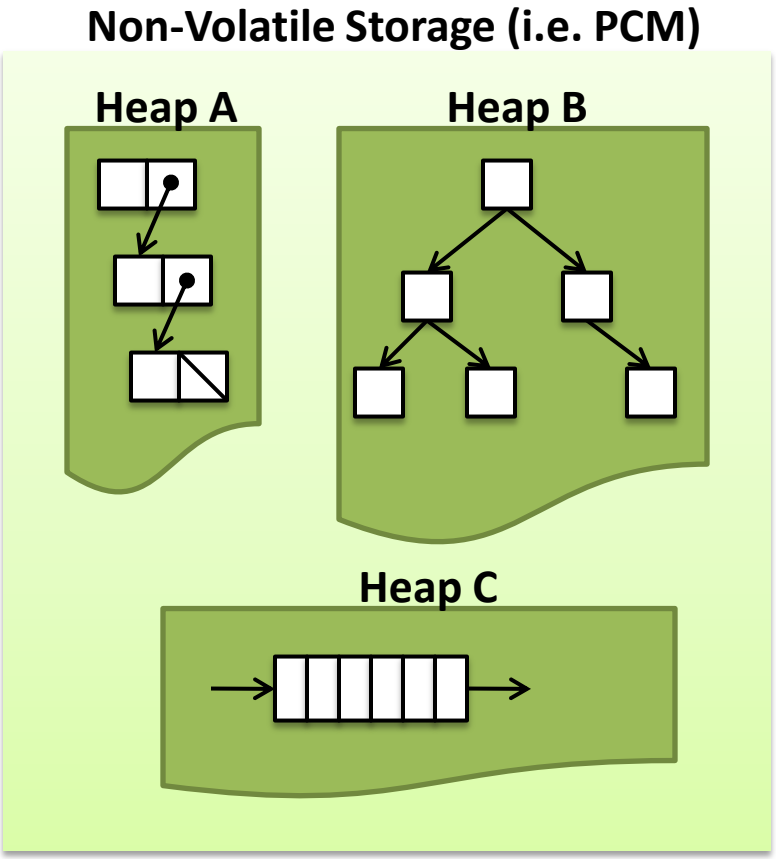
- A program accesses a single heap in volatile memory
  - Build rich, pointer-based data structures
- Serialize/de-serialize to transfer data to/from storage

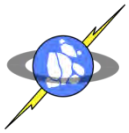




The New Way: Non-Volatile Heaps

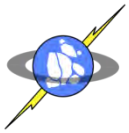
- We want to have multiple heaps in non-volatile storage
  - Appear in the application's address space just like memory
  - Rich, pointer-based data structures live directly in storage
  - Challenge: provide safety guarantees





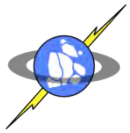
Accessing Non-Volatile Heaps: Transactions

- Used by databases to make guarantees about storage
- Now also used to manage concurrent access to volatile memory in multi-core systems
- Definition: a sequence of operations that is atomic and durable
- Give us guarantees against
  - Failures
  - Concurrency issues



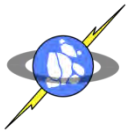
Transactions: How they work

1. Begin transaction
2. Perform operations
   - Record the state of any data that is read or written in a log to be played back later if the transaction fails
3. If no conflicts, then commit (makes permanent)
   - Conflict: data was changed by another transaction during execution
4. If conflicts, then abort and rollback (undo)
   - Replay the logs to restore old values
5. End transaction



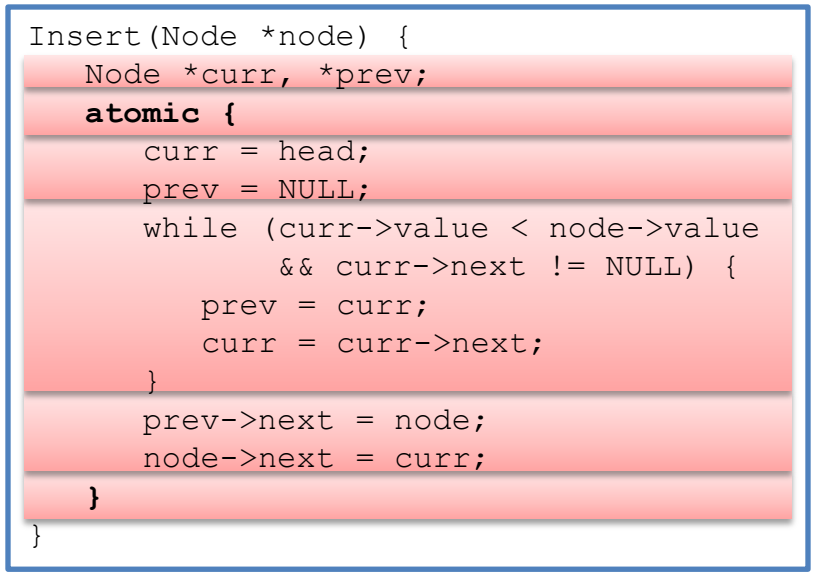
A Simple Example

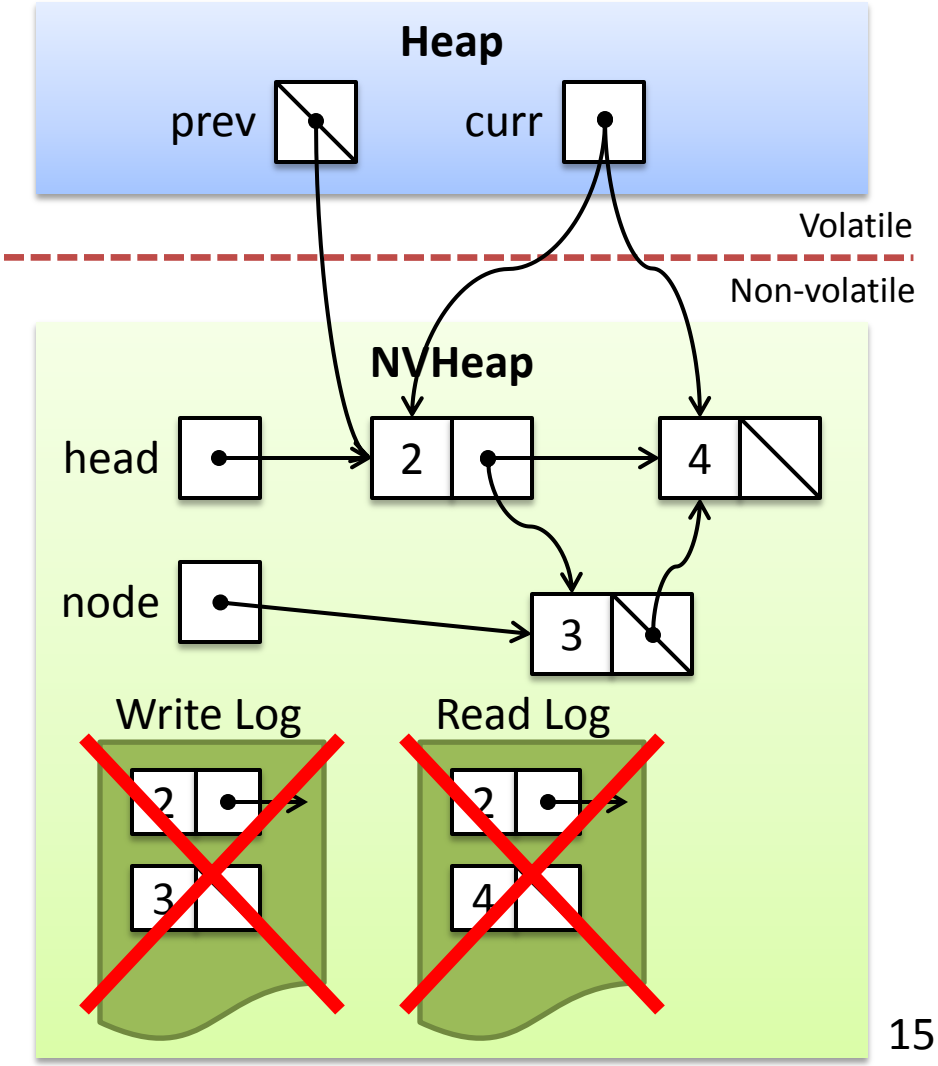
- We have a linked list in non-volatile storage
- We want to insert a node in sorted order
- Requirement: If we crash, our list must remain in a good state
  - Either the node is added or it is not
  - No wild pointers, missing links, etc.



A Simple Example

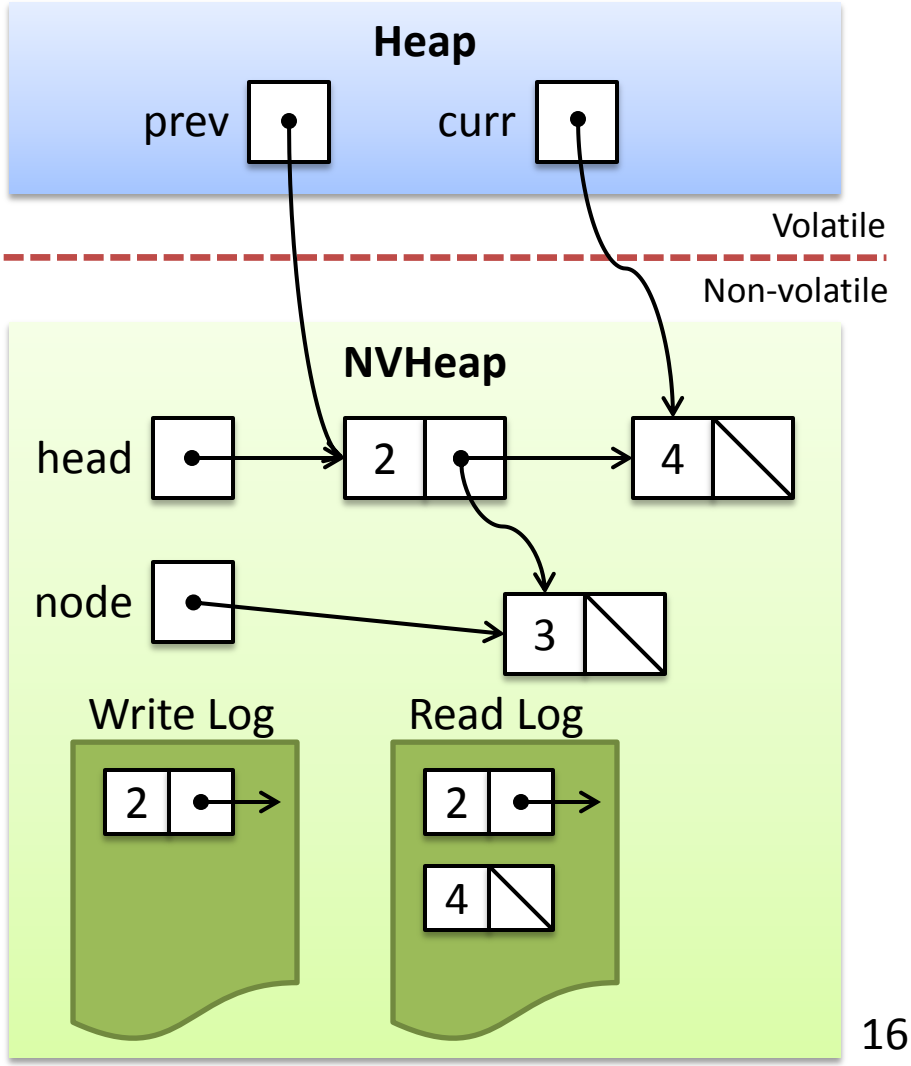
```
Insert(Node *node) {
   Node *curr, *prev;
   atomic {
      curr = head;
      prev = NULL;
      while (curr->value < node->value
             && curr->next != NULL) {
         prev = curr;
         curr = curr->next;
      }
      prev->next = node;
      node->next = curr;
   }
}
```





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

Conclusion

- New non-volatile memory technologies have the potential to revolutionize computing
  - DRAM-like performance (both sequential and random access) + density + persistence
- Re-designing these abstractions will unleash the full potential of these technologies!