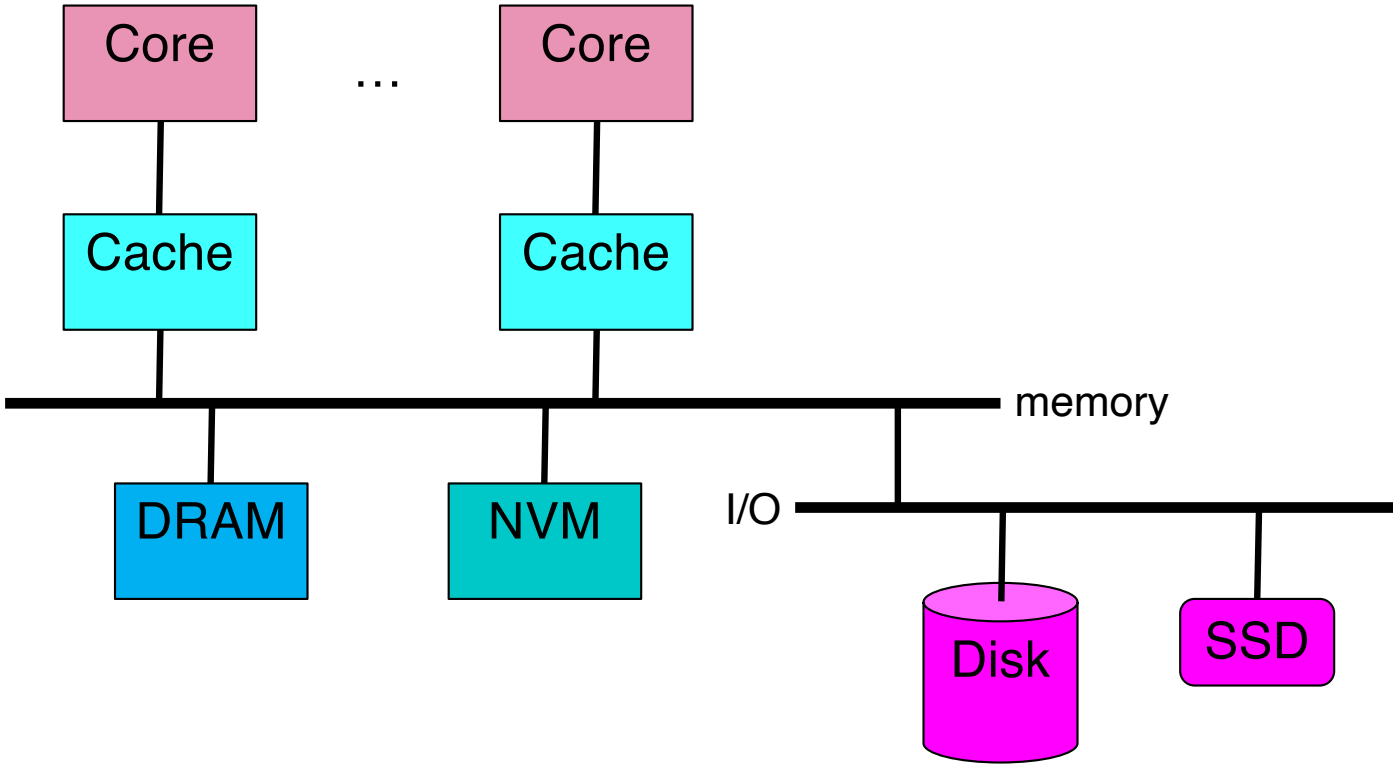
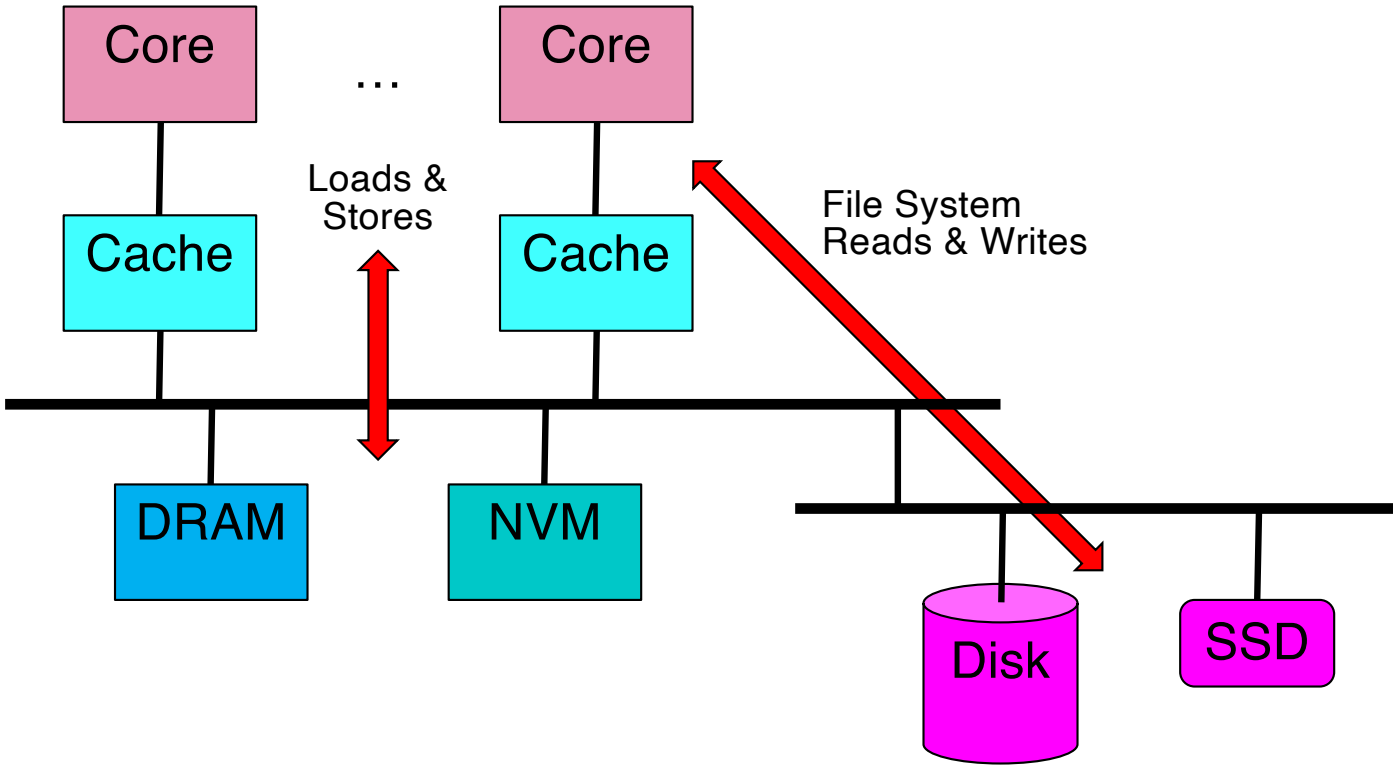
Fast Nonvolatile Memory

- NVM is on its way: PCM (Intel Optane), ReRAM, STT-MRAM, ...
  - » Could just treat it as dense, low-power DRAM replacement
  - » Tempting to put some long-lived data "in memory," rather than serializing to the file system
  - » (Could also consider full-system persistence — not the topic of this talk.)
- Raises issues of
  - » Correctness in the wake of a crash
  - » Safety with buggy or untrusted programs
  - » System design for persistent segments

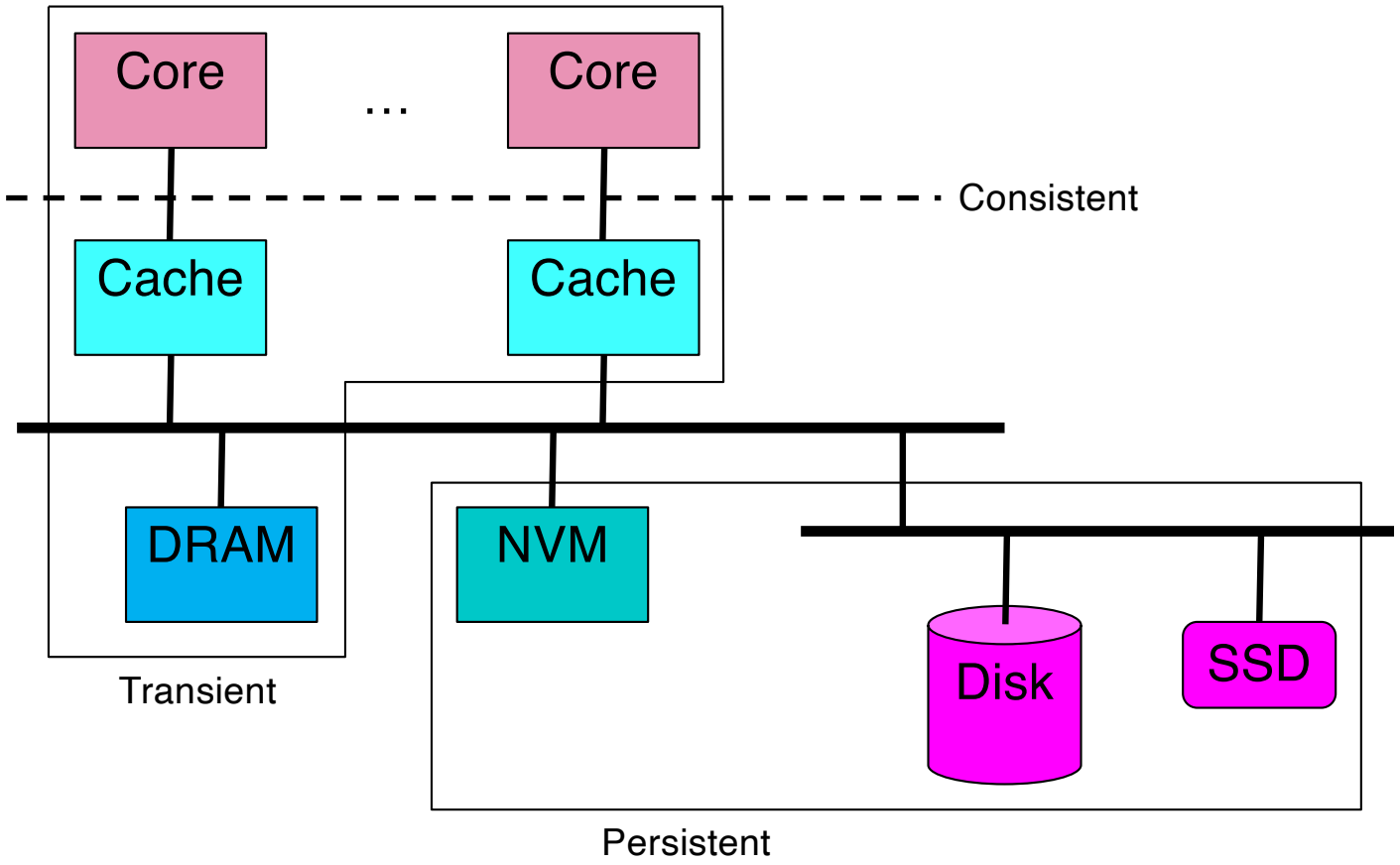
The Consistency Challenge



The Consistency Challenge



The Consistency Challenge



Out-of-Order Write-back

```
p = new node();
q->next = p;
```

- Danger that `q` will persist before `*p`
  - » Have to explicitly force data to memory in order
- Need formal models of correctness
  - » *durable linearizability, explicit epoch persistency*
- Need to map the program to the hardware
  - » Automatic transforms (too slow)
  - » Hand-tuned persistent data structures
  - » Design principles, libraries, and proof techniques

The Montage System

- *Buffered* durable linearizability
  - » On a crash, you may lose what you did very recently
  - » but what's left is consistent
  - » and you can force currency with synch()
- Basic idea
  - » slow-ticking epoch clock
  - » never the case that A happens before B but lies in an earlier epoch
  - » on a crash, recover to second-most-recent epoch boundary
- Blocking (lock-based – Haosen Wen) and nonblocking (Wentao Cai) versions

Persistent Memory Allocation

- Ralloc — Wentao Cai
- Goal: separate memory allocation from persistent data structure design
  - » allow a single allocator to be used across structures
  - » ensure that all and only the unreachable blocks are free in the wake of a crash
- Key insight: any correct transient allocator can be used if you can garbage collect during recovery
  - » a *nonblocking* allocator is especially useful
- Introduce the notion of *filter functions* for better-than-conservative collection

Failure Atomicity
(for composite operations)

- For lock-based code: *failure-atomic sections* (FASEs) delimited by outermost locks
  - » Roll incomplete work back during recovery (Undo logging)
  - » Roll incomplete work forward during recovery (Redo logging)
  - » *Execute* forward to end of critical section: JUSTDO/iDO
- Transactions (for concurrent, deadlock-free composition)
  - » Leverage past work with (transient) transactional memory

Nonblocking Persistent Transactions

- QSTM — Alan Beadle
  - » Based on the RingSTM of Spear & Michael
  - » Concurrent planning with one-at-a-time commit
- txMontage — Wentao Cai
  - » Extends nbMontage
  - » All operations of a given transaction occur in the same epoch and commit or abort together
- Lazy Persistent Medley — Mingzhe Du
  - » Don't resolve conflicts until you're ready to commit
  - » Eliminate livelock
  - » Achieve the benefits of buffering without an epoch system