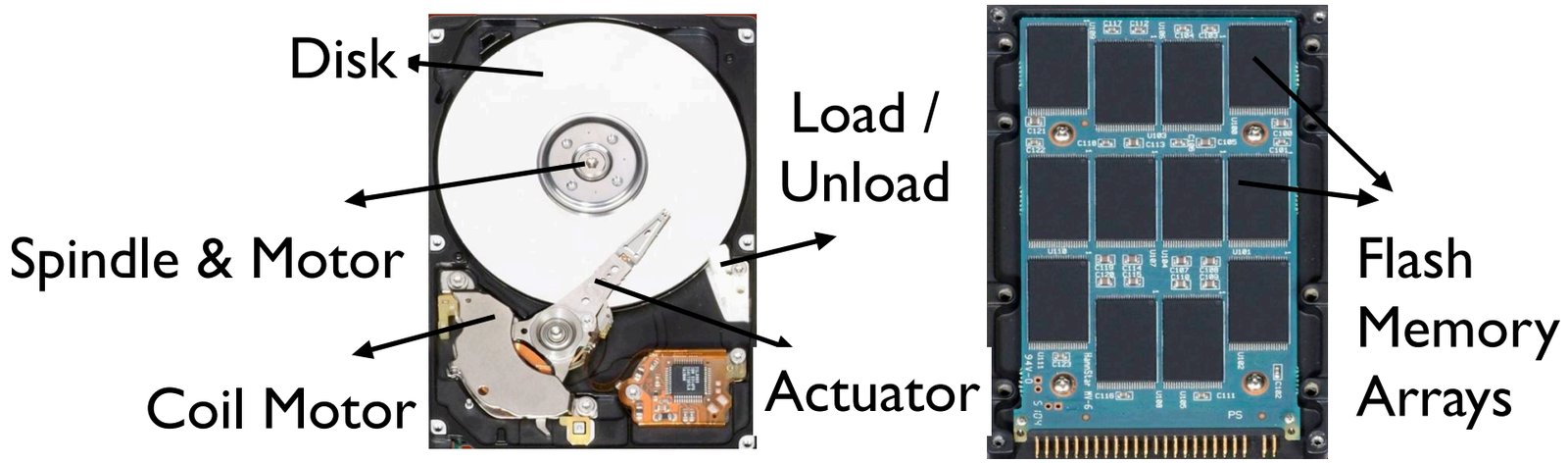
Performance of PC Solid-State Disks

as a Function of Bandwidth, Concurrency, Device Architecture, and System Organization

Cagdas Dirik & Bruce Jacob
Electrical & Computer Engineering
University of Maryland at College Park
cdirik@umd.edu

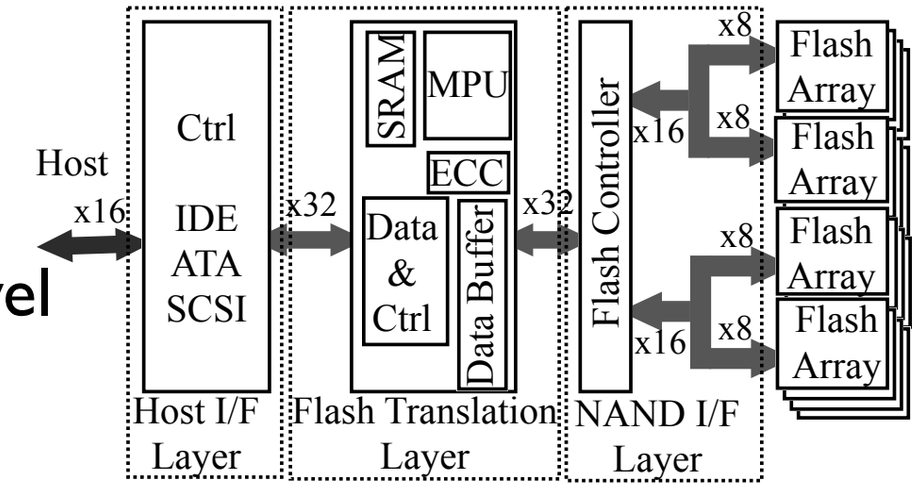
SSD Design is NOT Simple!

- Simple interface - no mechanics



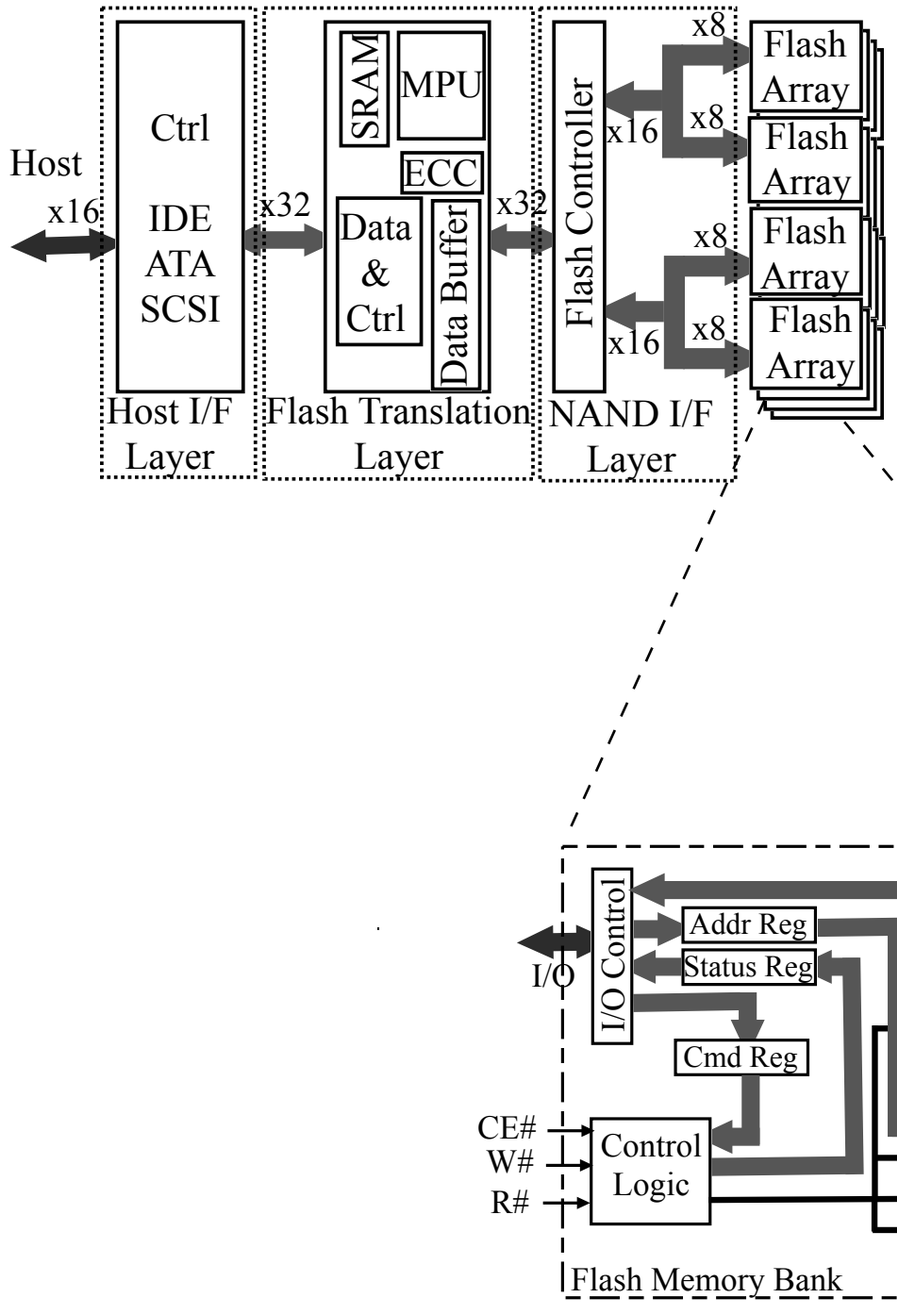
- No in-place update of data
- Asymmetric read and write + erase time
- Lack of models
  - Timing/performance
  - Power

SSD Design is NOT Simple!



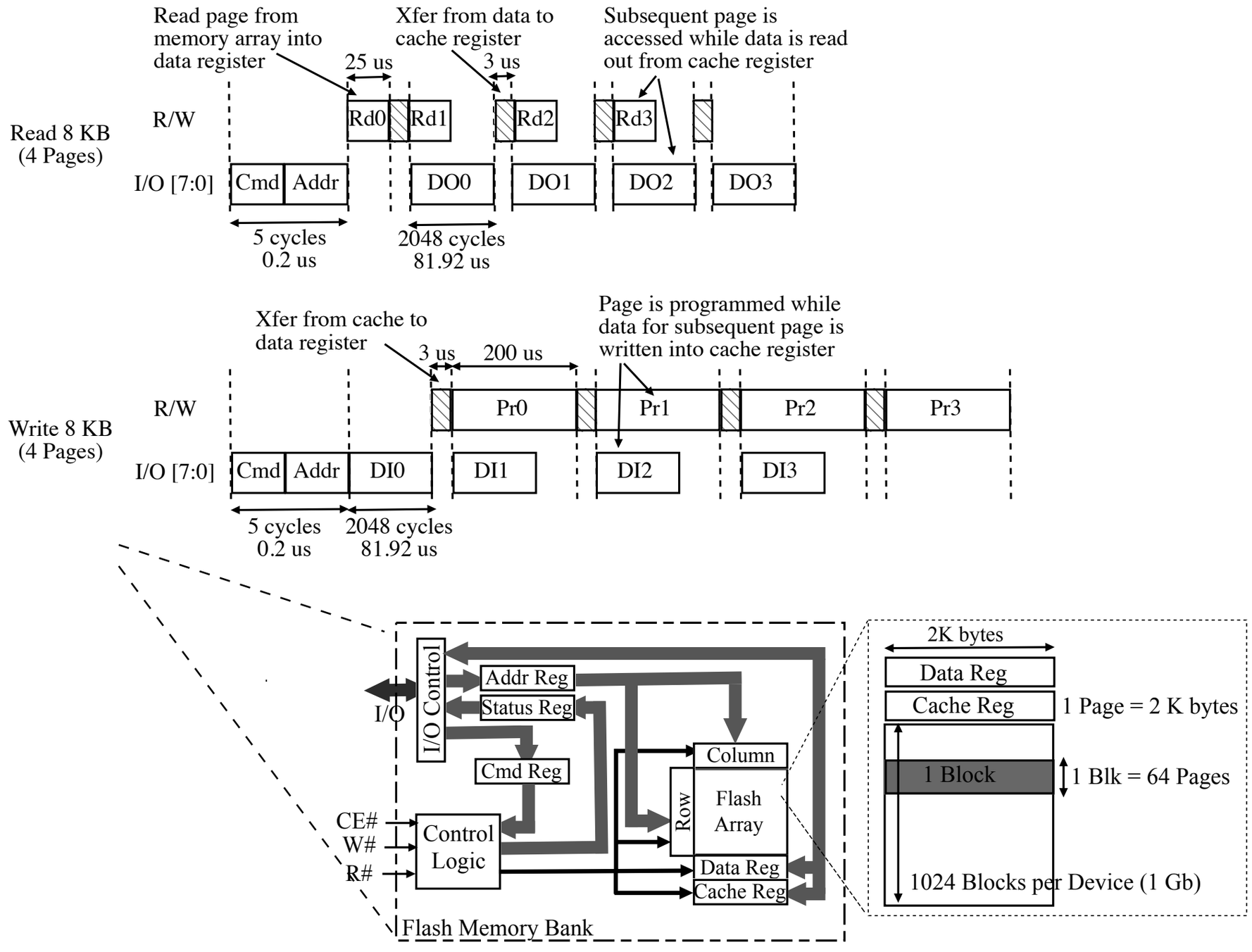
- Concurrency
  - System- & device-level
  - Architecture
  - I/O bandwidth utilization
- Proprietary firmware
  - FTL and/or Flash Controller
  - Mapping algorithms, wear leveling, garbage collection, ECC, request interleaving, write scheduling/striping
- Workload sensitive

Modeling SSDs



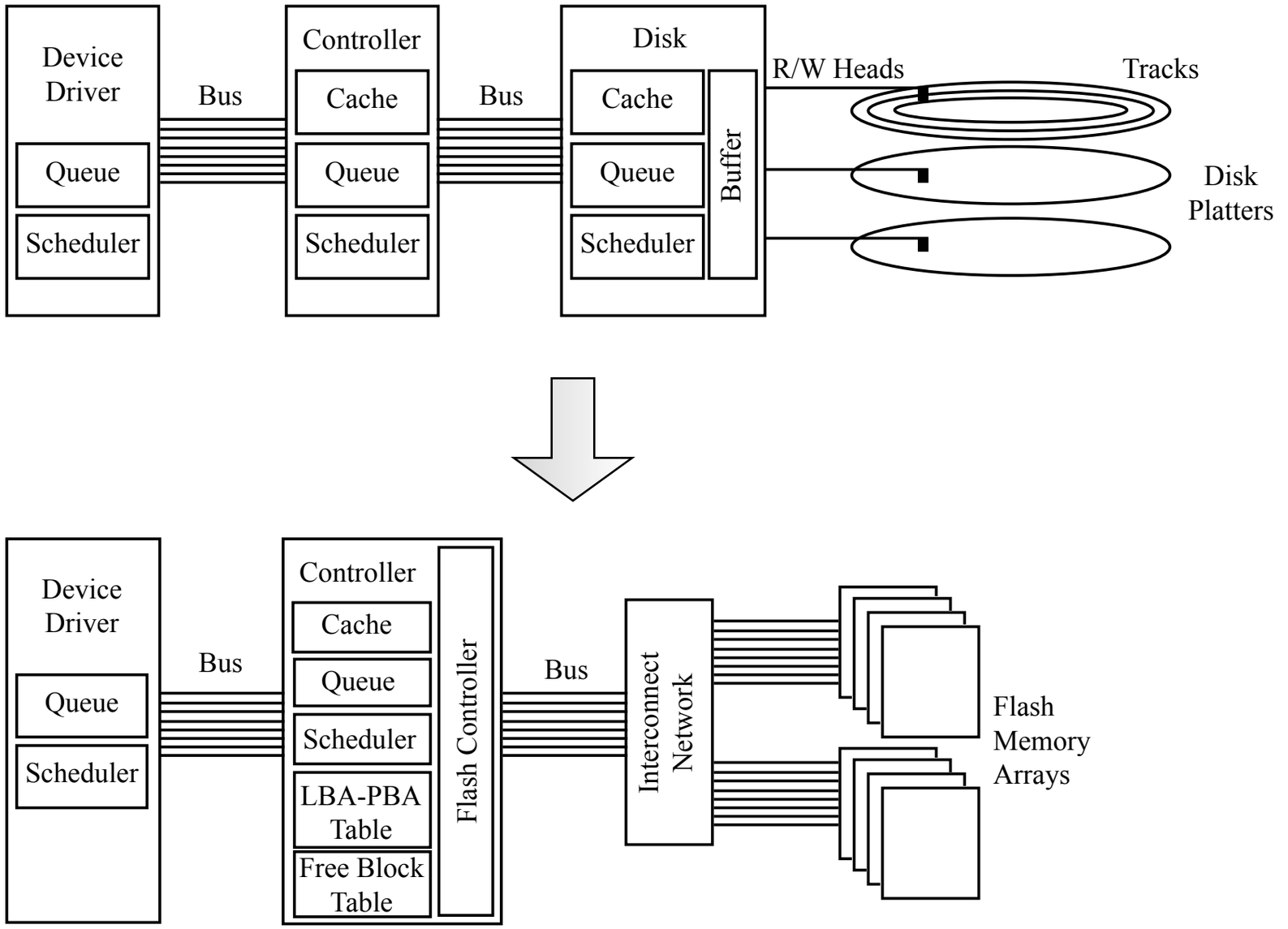
- 2 KB Page
- 128 KB Block
- 25 µs page read
- 200 µs page program
- 3 ms block erase
- 32 GB total storage

Modeling SSDs

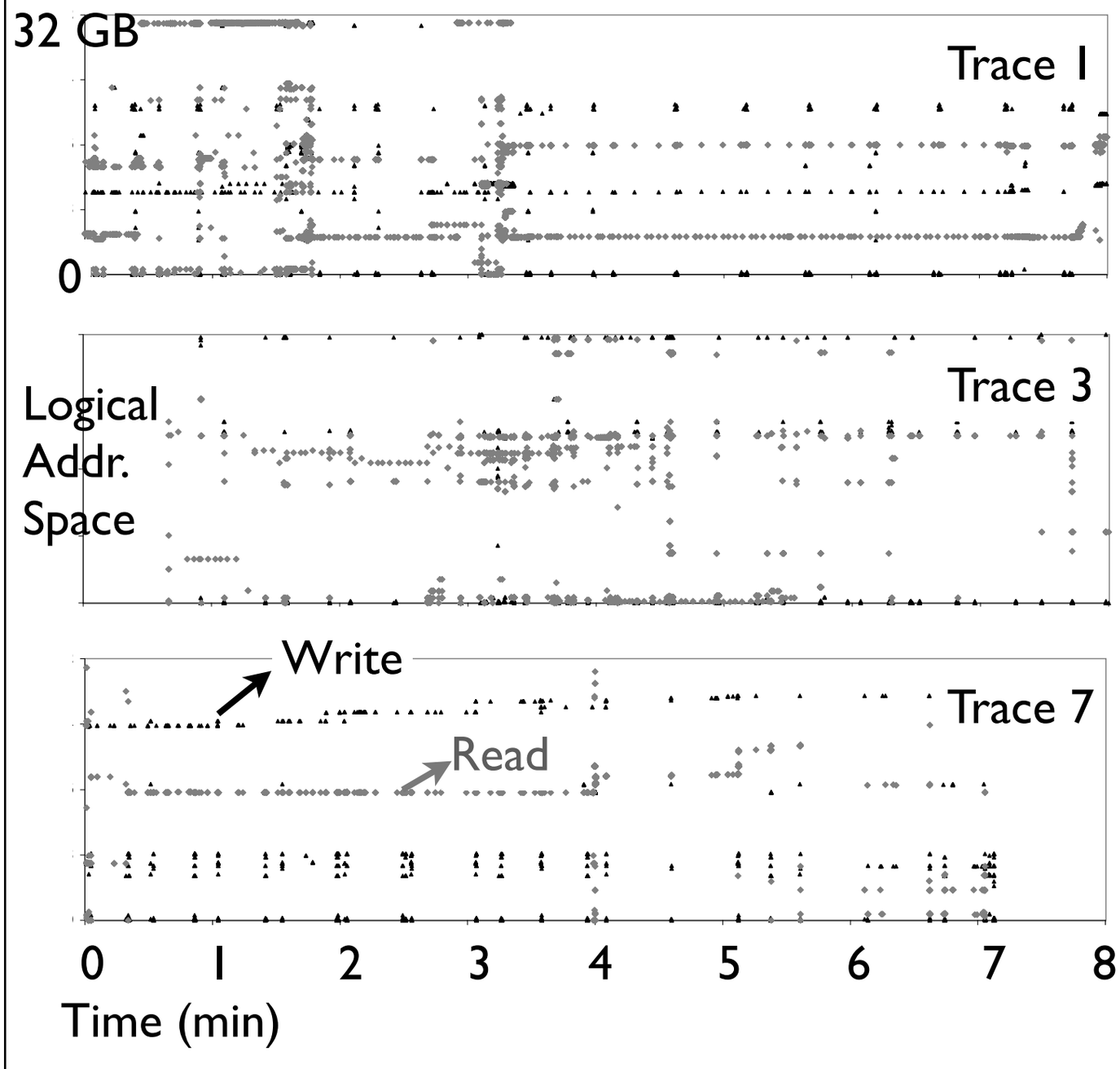


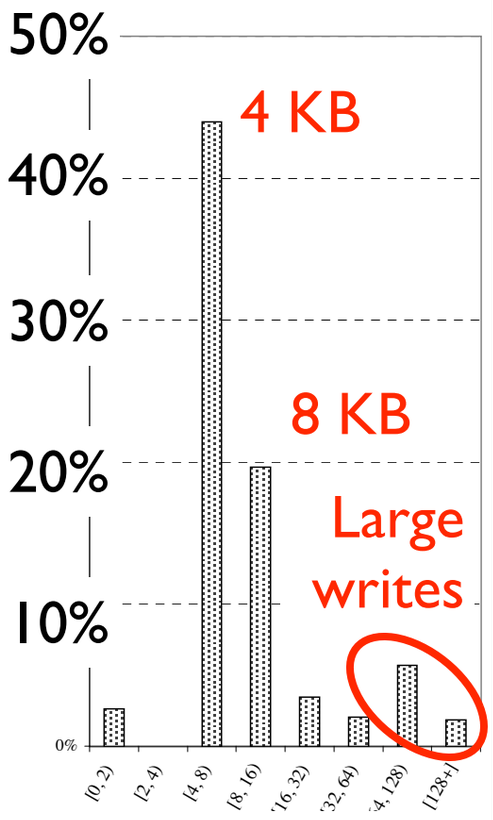
SSD Simulator

Based on DiskSim v2.0 [Ganger99]



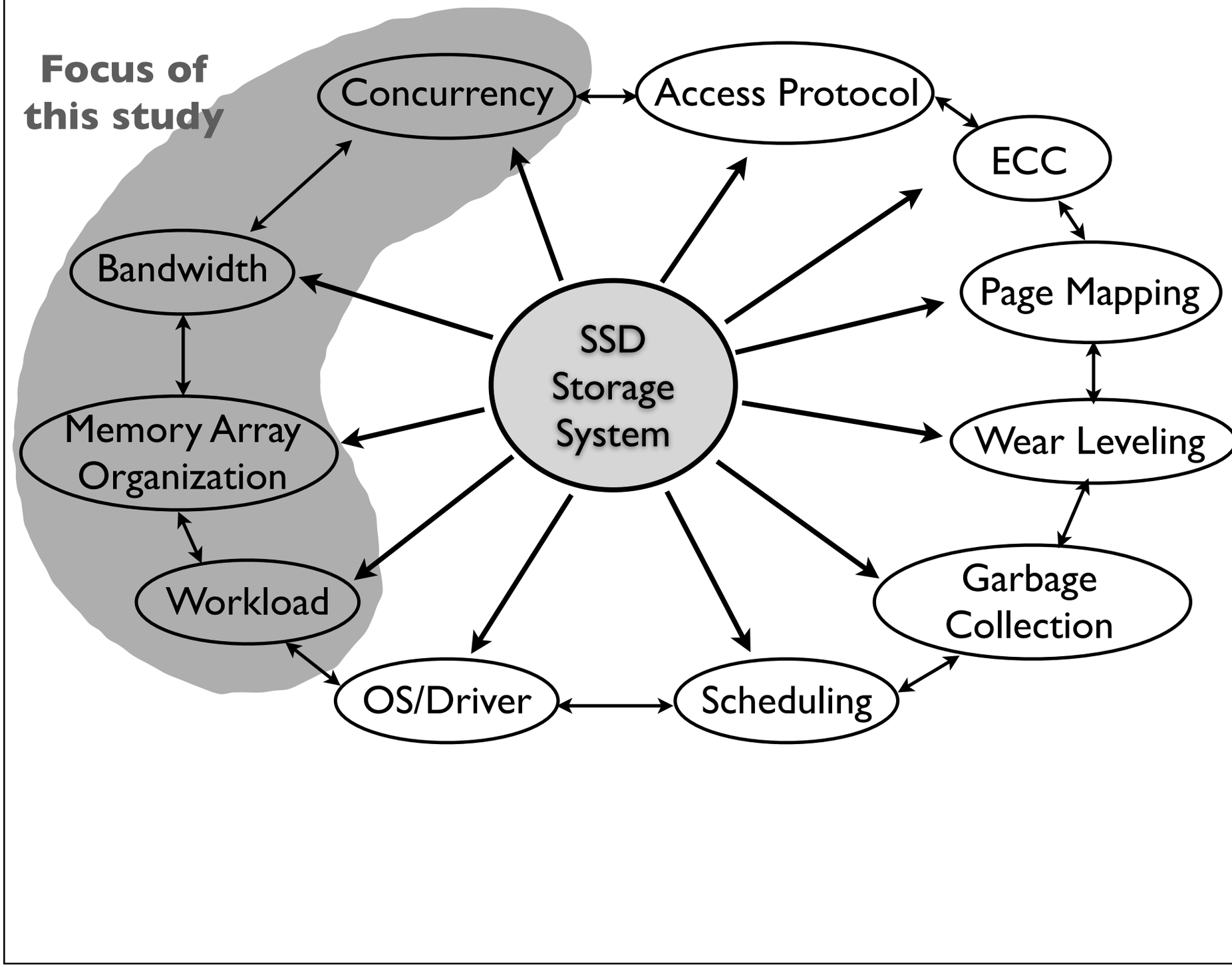
PC User Workloads





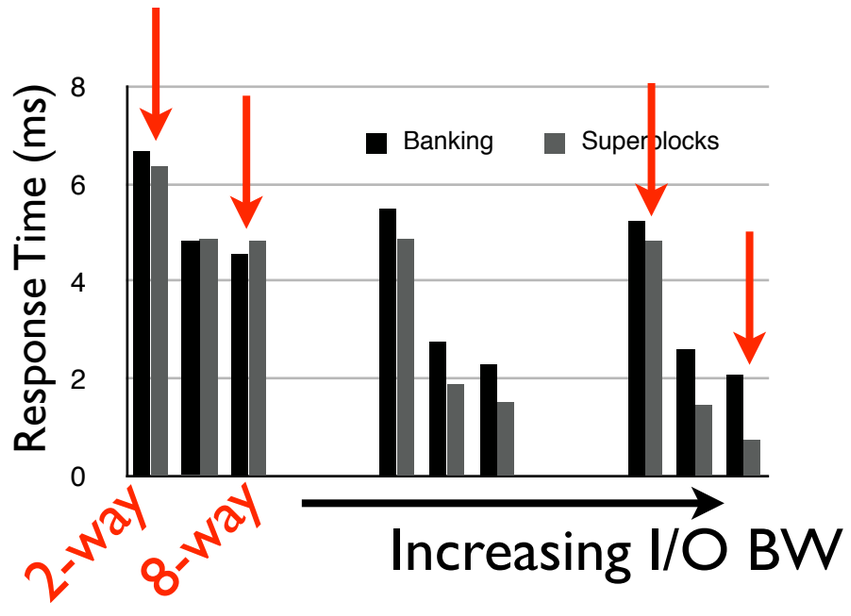
Request Size Distr.

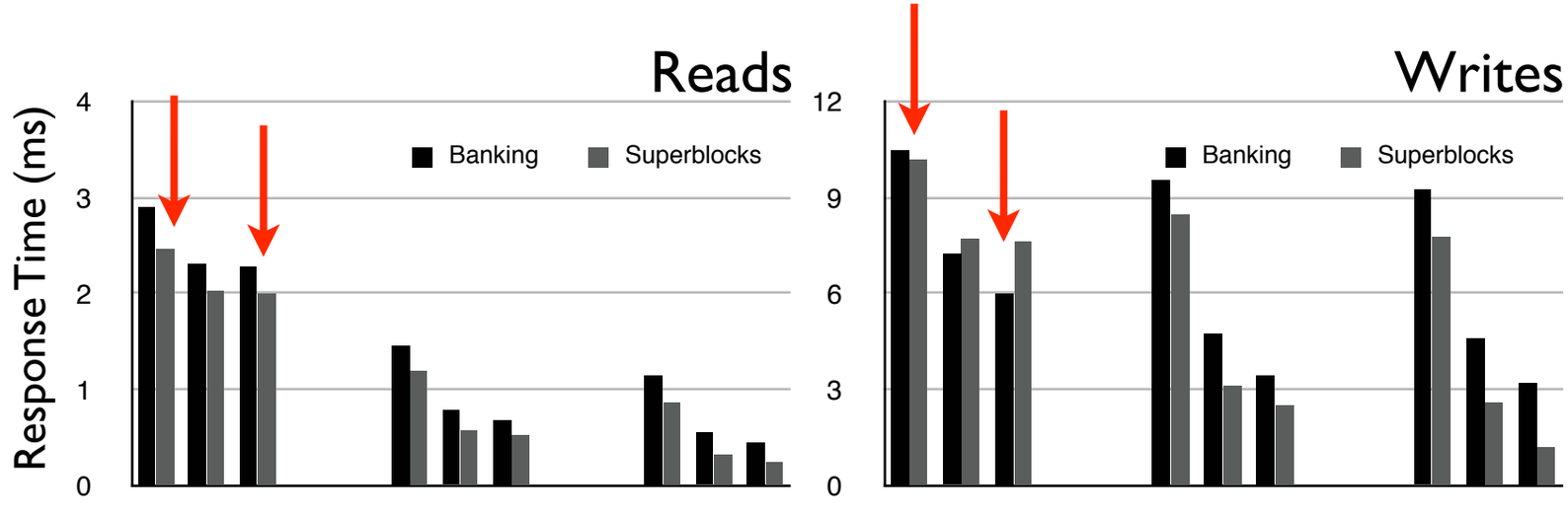
Design Space



Concurrency

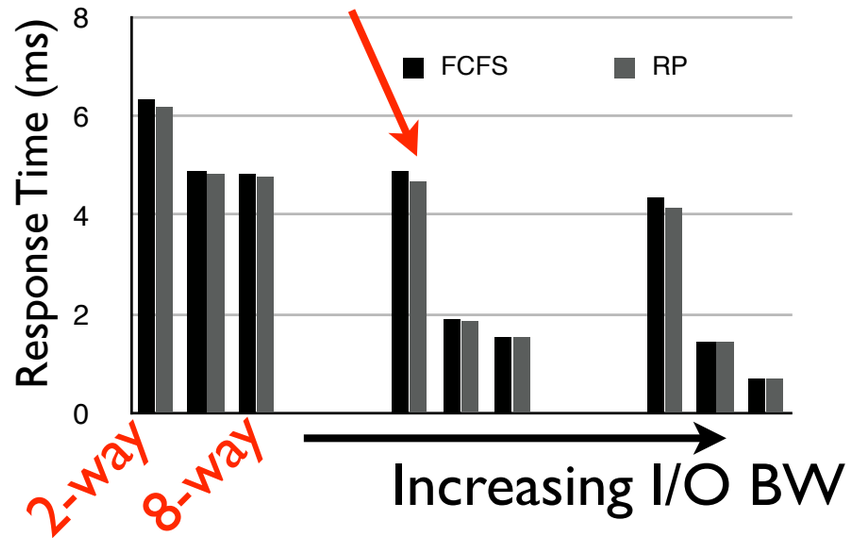
Banking vs. Superblocks vs. Bandwidth

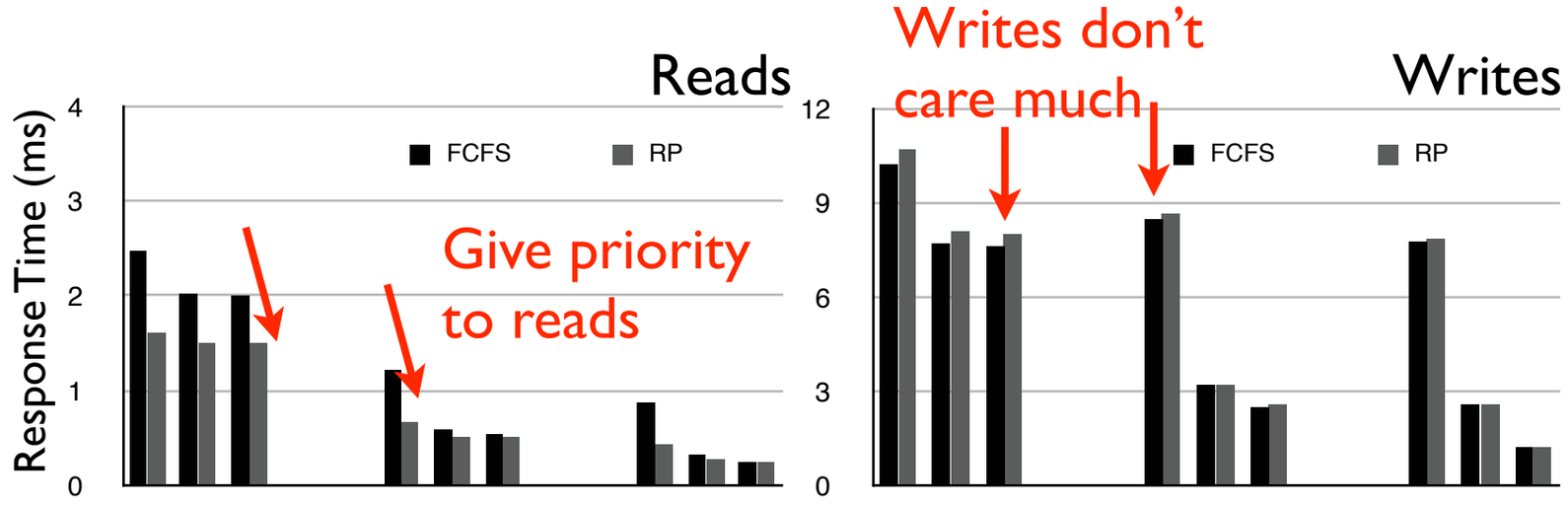




Request Scheduling

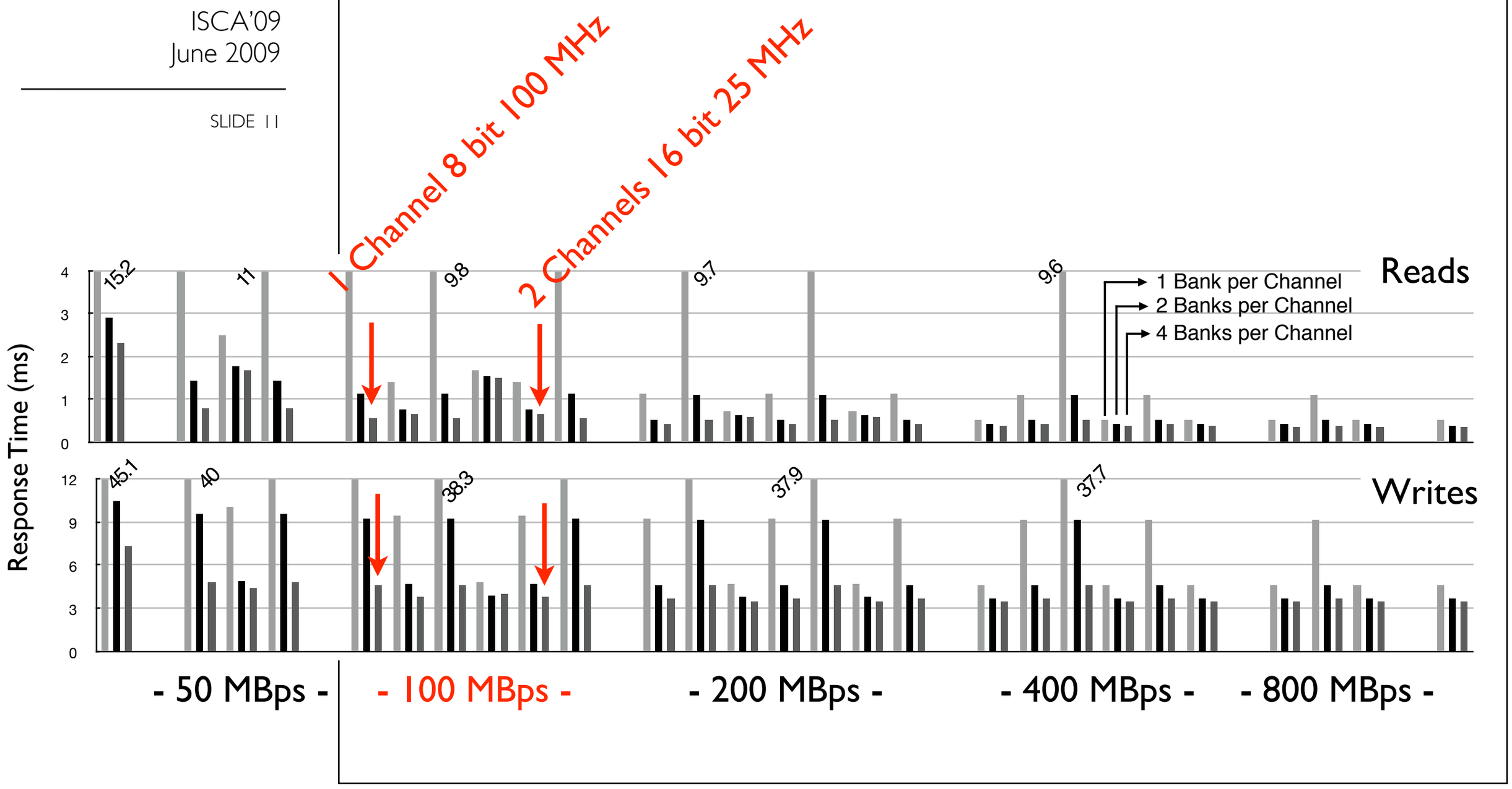
What if we give priority to reads?





SSD Organization

Several near optimal configurations



Bottom Line

- Design of SSDs is a complex problem
- No simple solution as:
  - Increase I/O bandwidth
  - Increase parallelism
- **We should model and explore the design space!**
  - Timing and Power: SLC, MLC, ECC
  - Massive parallelism: 64, 128, 256 banks
  - FTL algorithms: Mapping, ECC, GC, Scheduling
  - Application specific design: DB, Web Server, etc.