Attention please!

- An intermediate report that summarizes what you have done in research project (5%) (2 pages in IEEE 2-column format, Due on April 2 in class).
- Please read the first 12 pages of the article about the Byzantine generals problem sent to you by the grader.
- Our Midterm Exam is scheduled on April 18 in class.

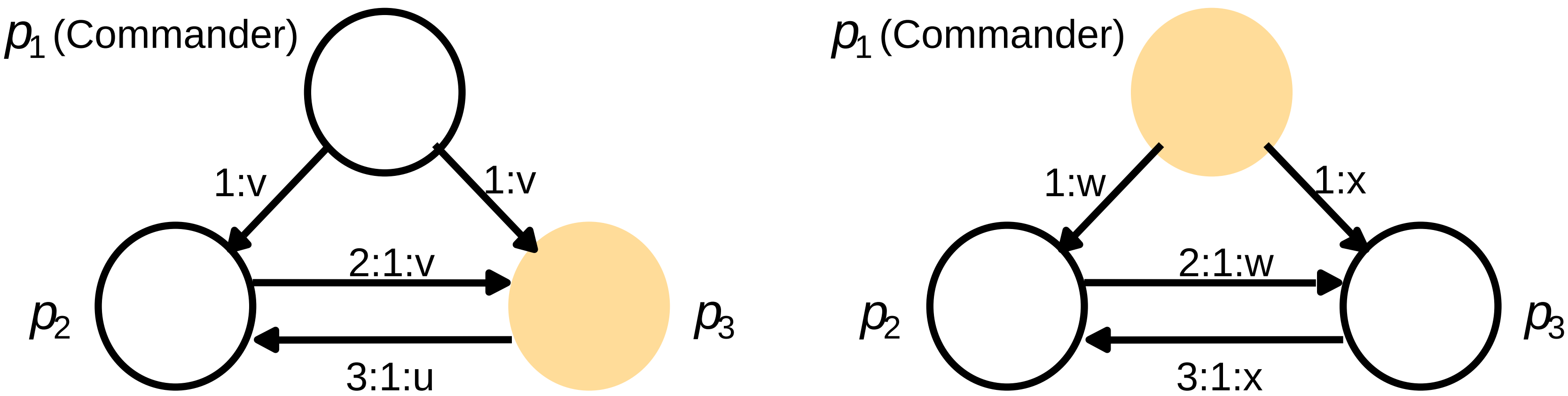
Byzantine generals problem in a synchronous system

- Processes can exhibit arbitrary failures
- Up to *f* of the *N* processes may faulty
- Correct processes can detect the absence of a message through a timeout
- Communication channels between pairs of processes are private
- No faulty process can inject messages into the communication channel between correct processes

Impossibility with three processes

- Unsigned messages to one another
- Just one of the 3 processes is faulty
- No solution

3 Byzantine generals



Faulty processes are shown coloured

Consistent with our intuition

- Nothing can be done to improve a correct general's knowledge where it cannot tell which process is faulty

- Byzantine agreement can be reached for 3 generals, with one of them faulty, if the generals digitally sign their messages (Please read that article sent to you by the grader)

Impossibility with $N \leq 3f$

- You can show that there is no solution to the problem if the number of processes and the number of faulty processes satisfies

$$N \leq 3f$$

- You do this by taking a supposed solution with a more than a third of the processes faulty. And then turn this into a solution for 1 faulty and 2 correctly working generals, by getting the three generals to simulate the solution for more then 3 situation by passing more messages.

For more information about Byzantine

- Please read Lamport et al. paper that was emailed to you this morning.

- "The Byzantine Generals Problem", ACM Transactions on Programming Languages and Systems, Vol.4, No.3, pp. 382-401,July 1982.

Solution with one faulty process

- For simplicity we only consider N=4 and f=1 case

- The correct generals reach agreement in two rounds of messages:

1. In the first round, the commander sends a value to each of the lieutenants

2. In the second round, each of the lieutenants sends the value it received to its peers

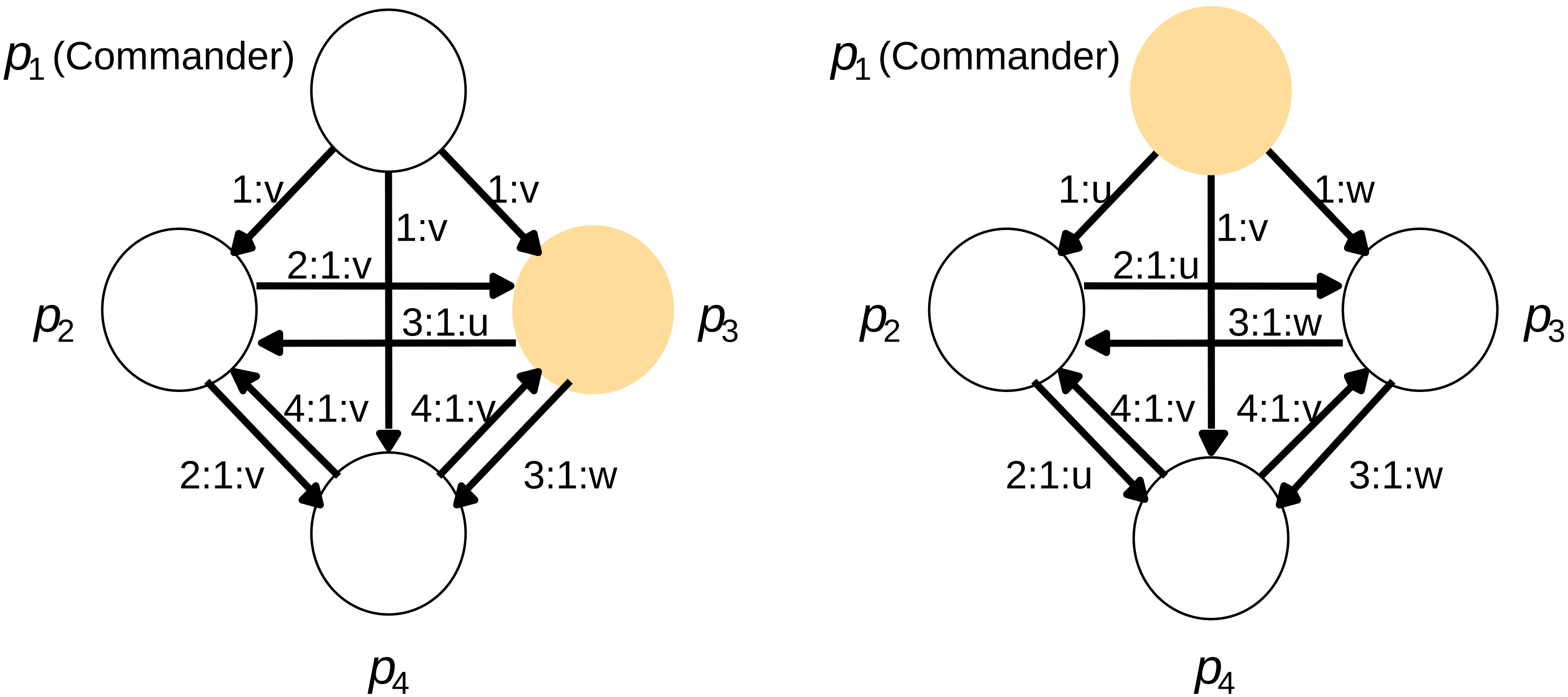
Algorithm description (1)

- A lieutenant receives a value from the commander, plus N-2 values from its peers
- If the commander is faulty, all the lieutenants have gathered exactly the set of values that the commander sent out
- If one of the lieutenants is faulty, each of its correct peers receives N-2 copies of the value that the commander sent, plus a value that the faulty lieutenant sent to it
- In either case, the correct lieutenants need only apply a simple majority function to the set of values that receive

Algorithm description (2)

- The correct lieutenants apply a simple majority function to the set of values they receive to get the correct value
- If there is no majority the majority function will return ?
- Also handles faulty processes that omit to send a message with ?
- In the general case ($f \geq 1$) operates over f+1 rounds
- Costly algorithm in terms of number of messages $O(N^{f+1})$

Four byzantine generals



Faulty processes are shown coloured

Left-hand case

- P_2 decides on *majority*(v,u,v)=v
- P_4 decides on *majority*(v,v,w)=v

Right-hand case

- P_2, P_3, and P_4 decide on *majority*(u,v,w)=?

Impossibility in asynchronous system

- The discussion so far has relied on message passing being synchronous
- Messages pass in rounds
- In an asynchronous system you can't distinguish between a late message and a faulty process
- No solution can guarantee to reach consensus in an asynchronous system

Assignment#2 (Chapter 15)

- 15.2

- 15.9

Sample Questions (1)

- ___F___ The faults classified as omission failures refer to cases when an component not only behaves erroneously, but also fails to behave consistently when interacting with multiple other components.
- ___T___ IP packets are addressed to computers, ports belong to the TCP and UDP level.
- ___F___ If a correct process issues *multicast(g,m)* and then *multicast(g,m')*, then every correct process that delivers m' will deliver m before m'. This is called total ordering.

Sample Questions (2)

- Lock timeouts can be used to resolve deadlocks. However, it has the following problems: a, b, c,

(a) Locks may be broken when there is no deadlock.

(b) If the system is overloaded, lock timeouts will happen more often and long transactions will be penalized.

(c) It is hard to select a suitable length for a timeout.

(d)Locks must be broken when there is a deadlock.

Sample Questions (3)

- In Chandy and Lamport 'snapshot' algorithm for determining global states of distributed systems, which of the following items were assumed?
  - (a) Neither channels nor process fail   a, b, c, d
  - (b) Any process may initiate a global snapshot at any time
  - (c) The graph of processes and channels is strongly connected
  - (d) The processes may continue their execution and send and receive normal messages while the snapshot takes place

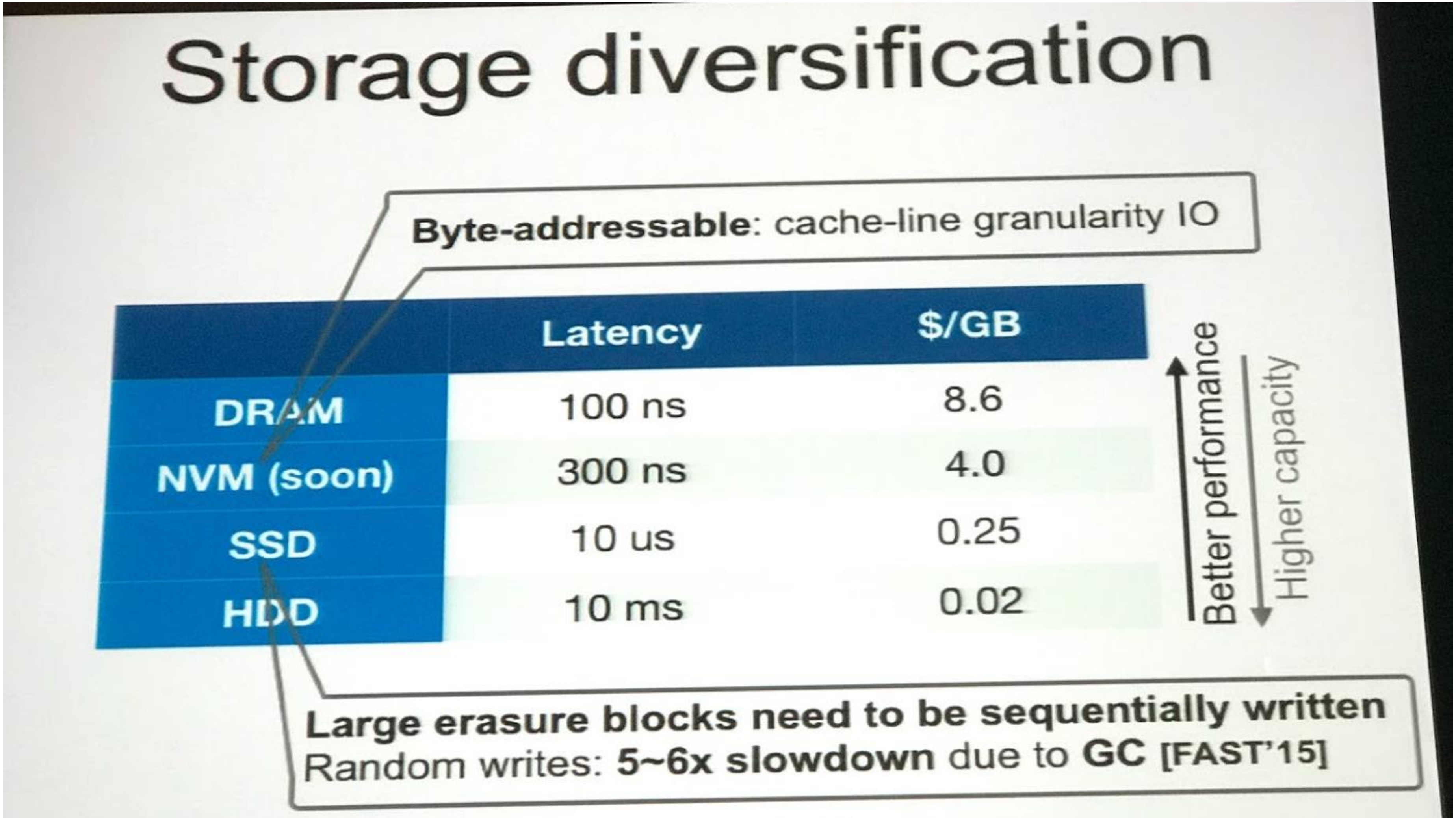
Sample Questions (4)

- The definition of uniform agreement is c
  - (a) If a correct process delivers message m, then all correct processes in group(m) will eventually deliver m.
  - (b) All correct processes in a group agree on a value.
  - (c) If a process, whether it is correct or fails, delivers message m, then all correct processes in group(m) will eventually deliver m.
  - (d) All processes in a group, whether they are correct or fail, agree on a value.

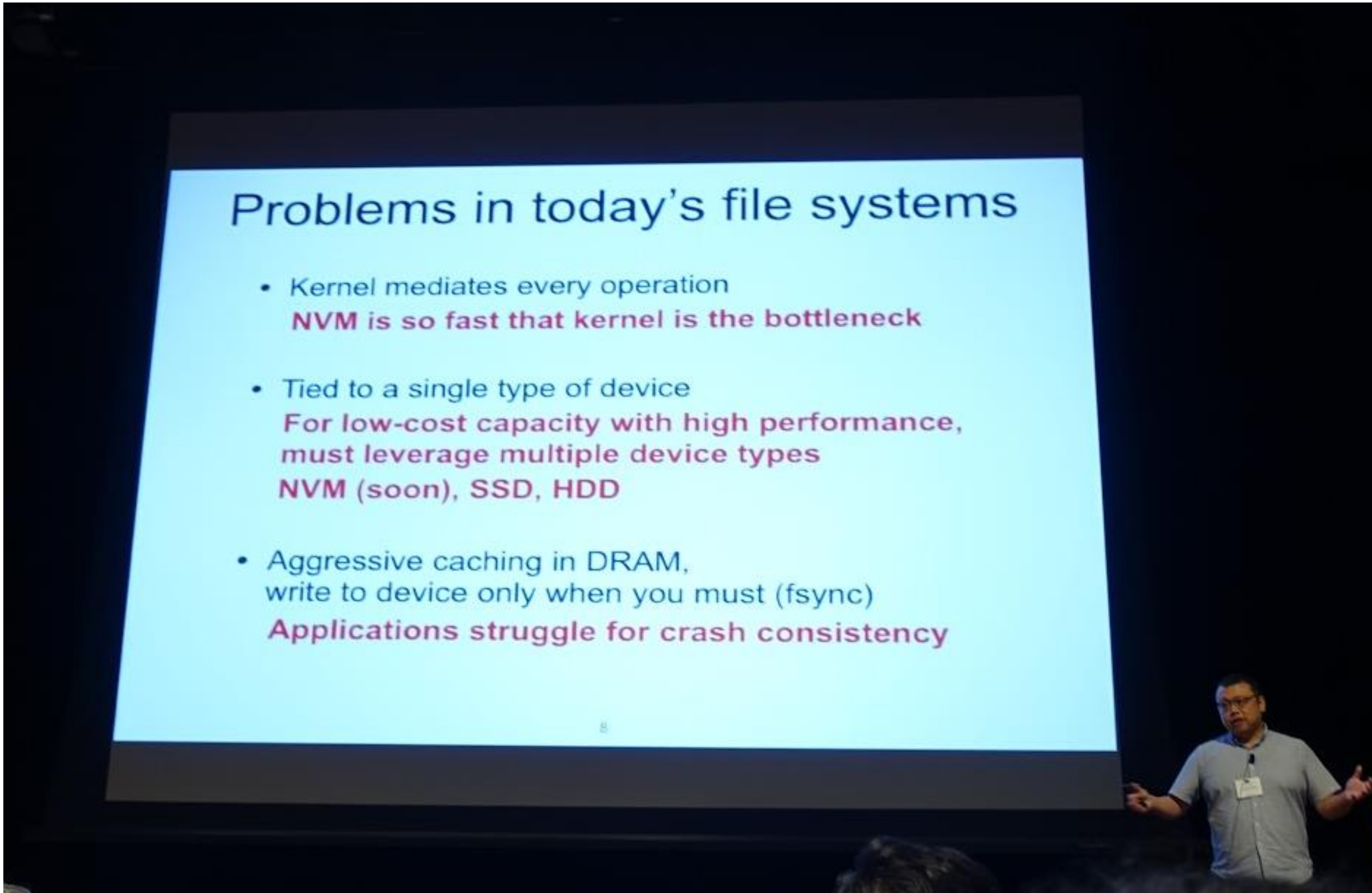
Sample Questions (5)

- Understand the byzantine generals problem.
- Understand the three choices of RMI invocation semantics.
- Understand the logical time vector.

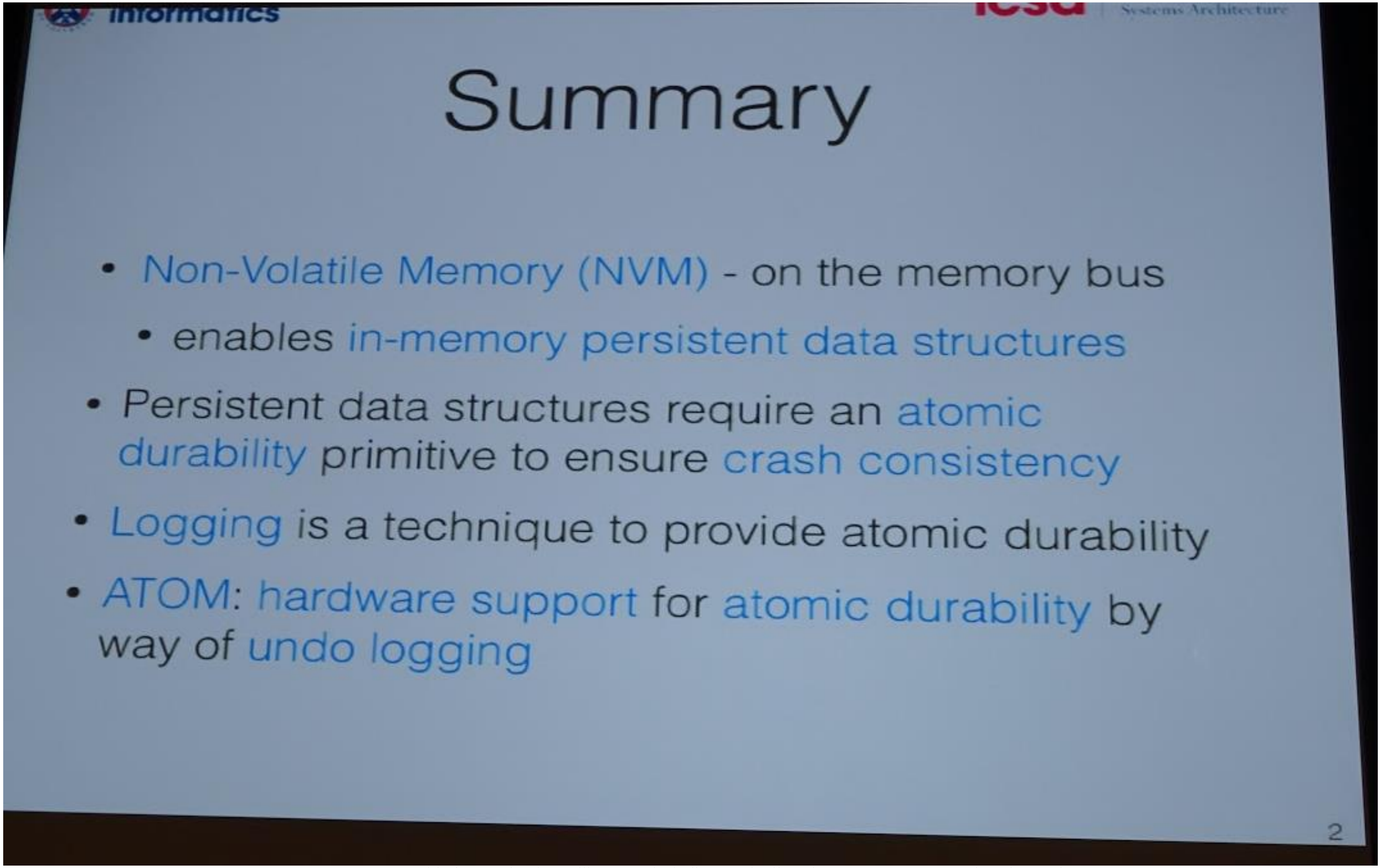
From NVMW 2018



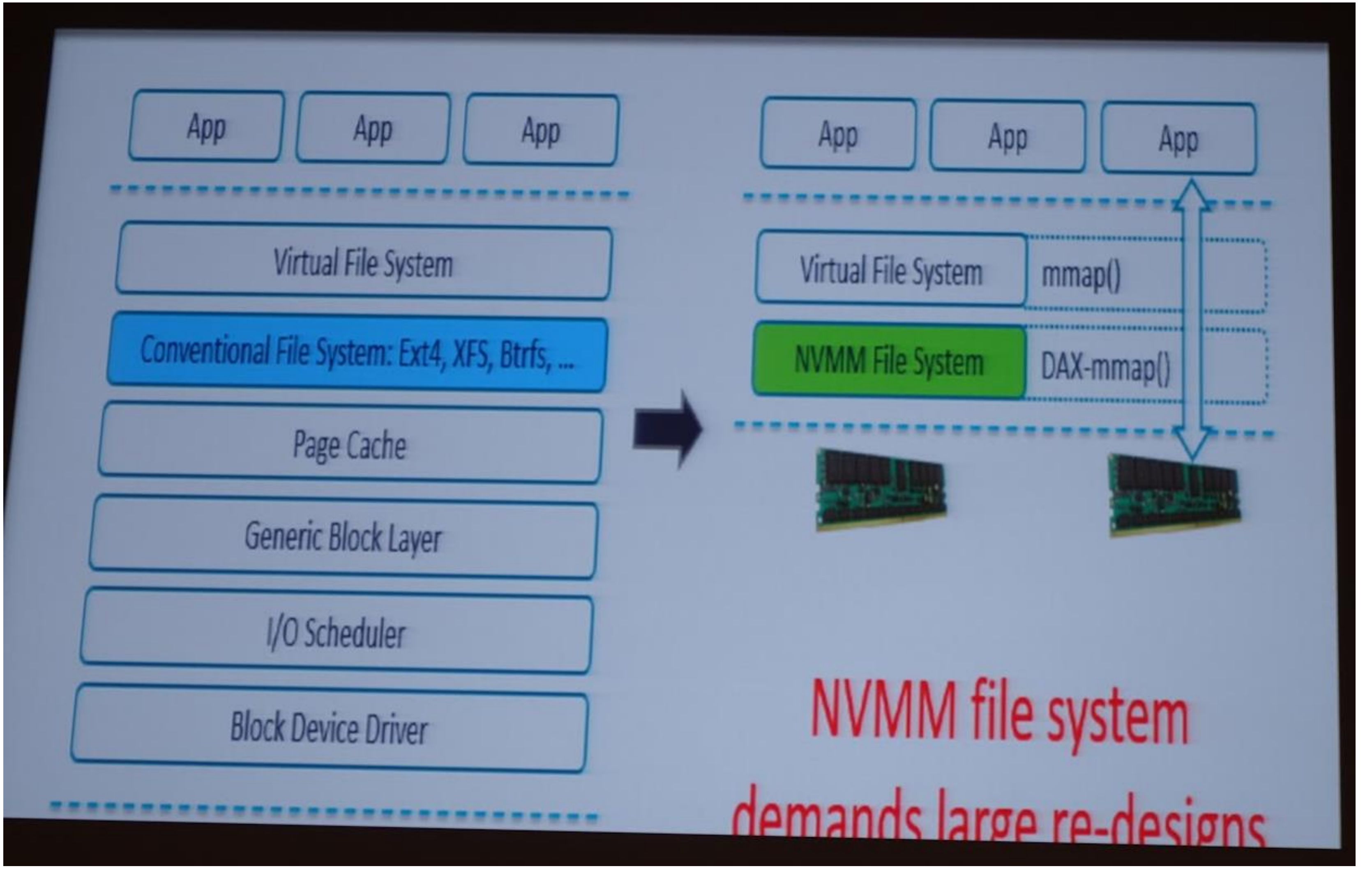
From NVMW 2018



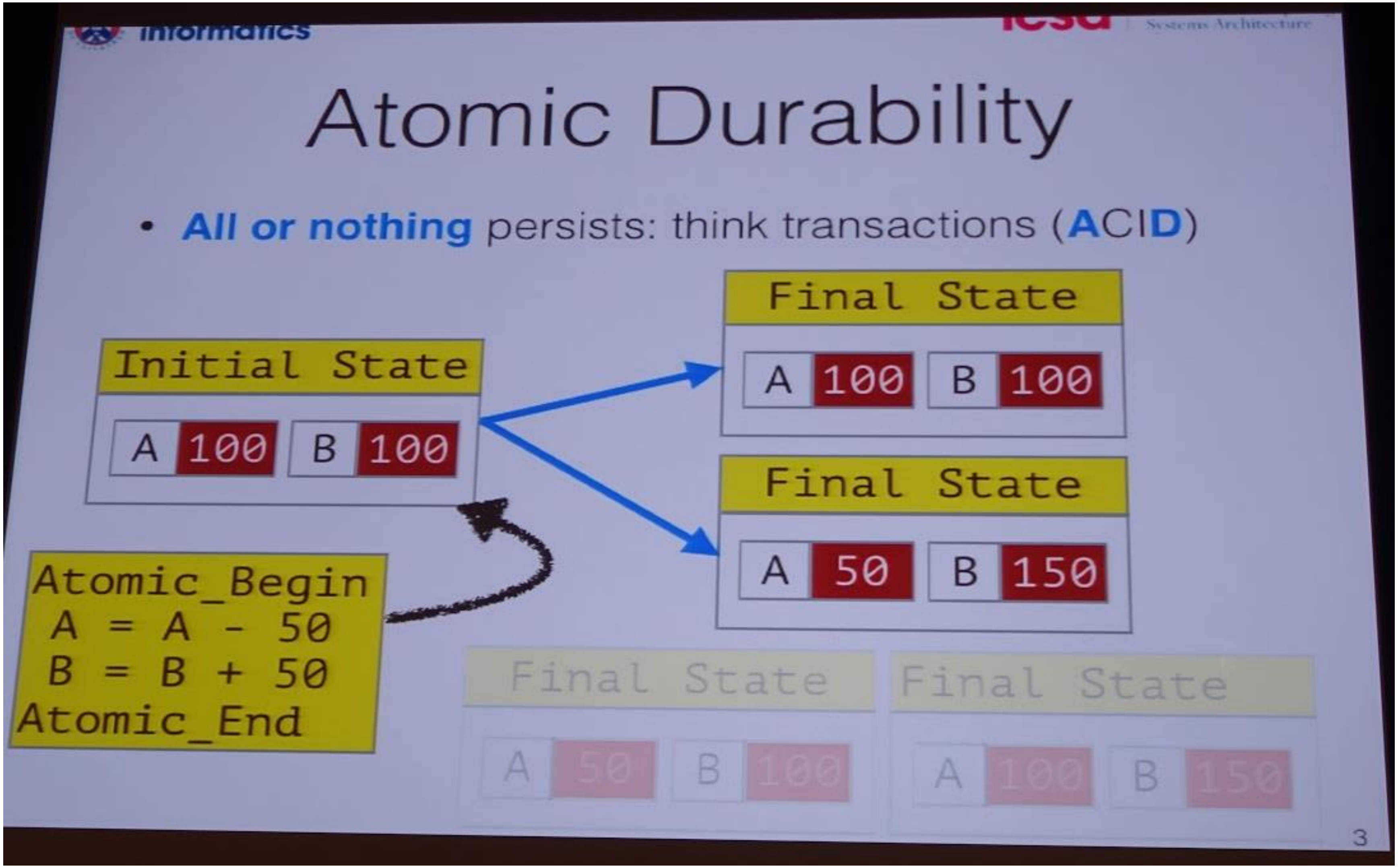
From NVMW 2018



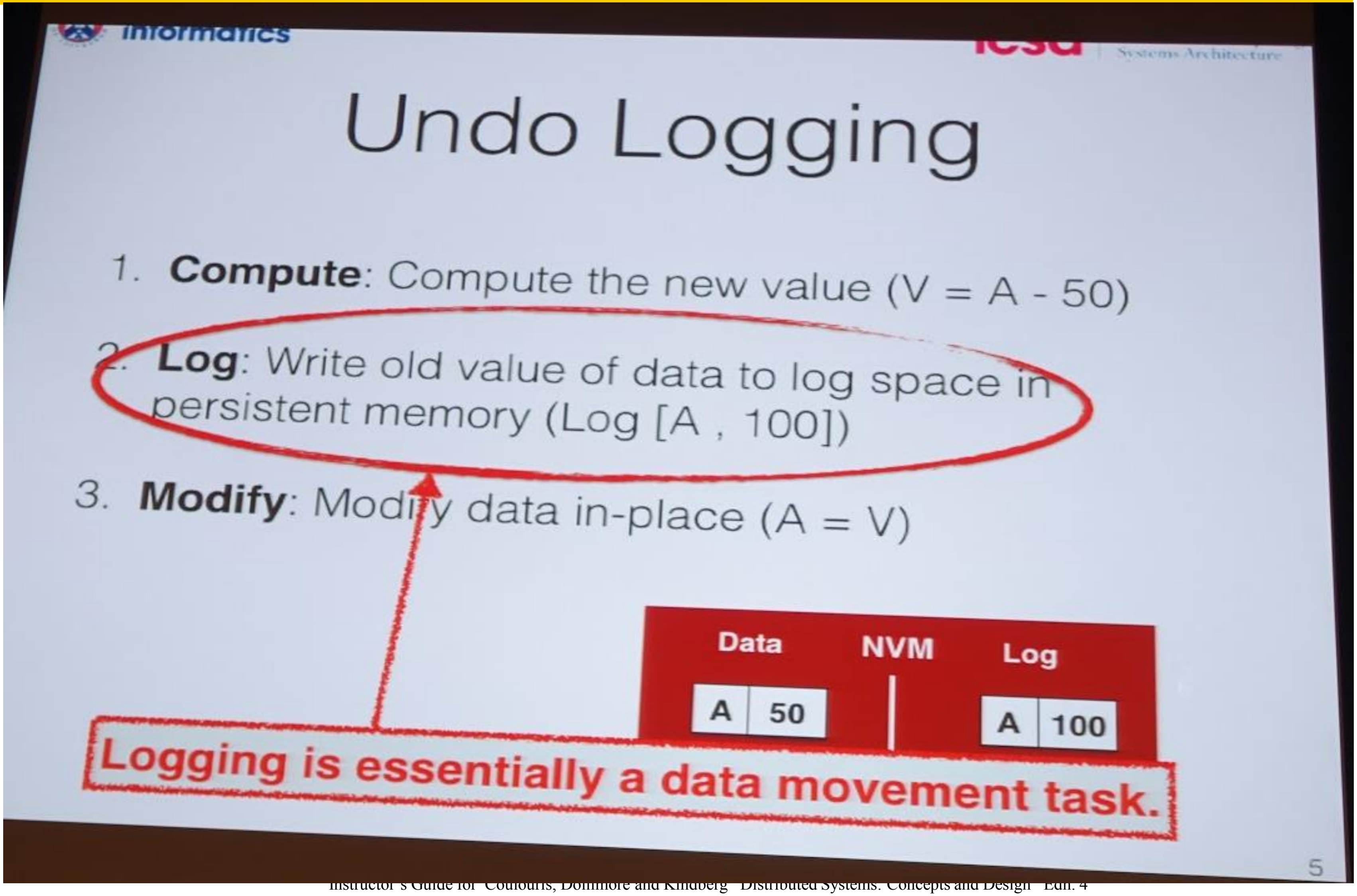
From NVMW 2018



From NVMW 2018



From NVMW 2018



For more information of NVMW 2018

- http://nvmw.ucsd.edu/program/