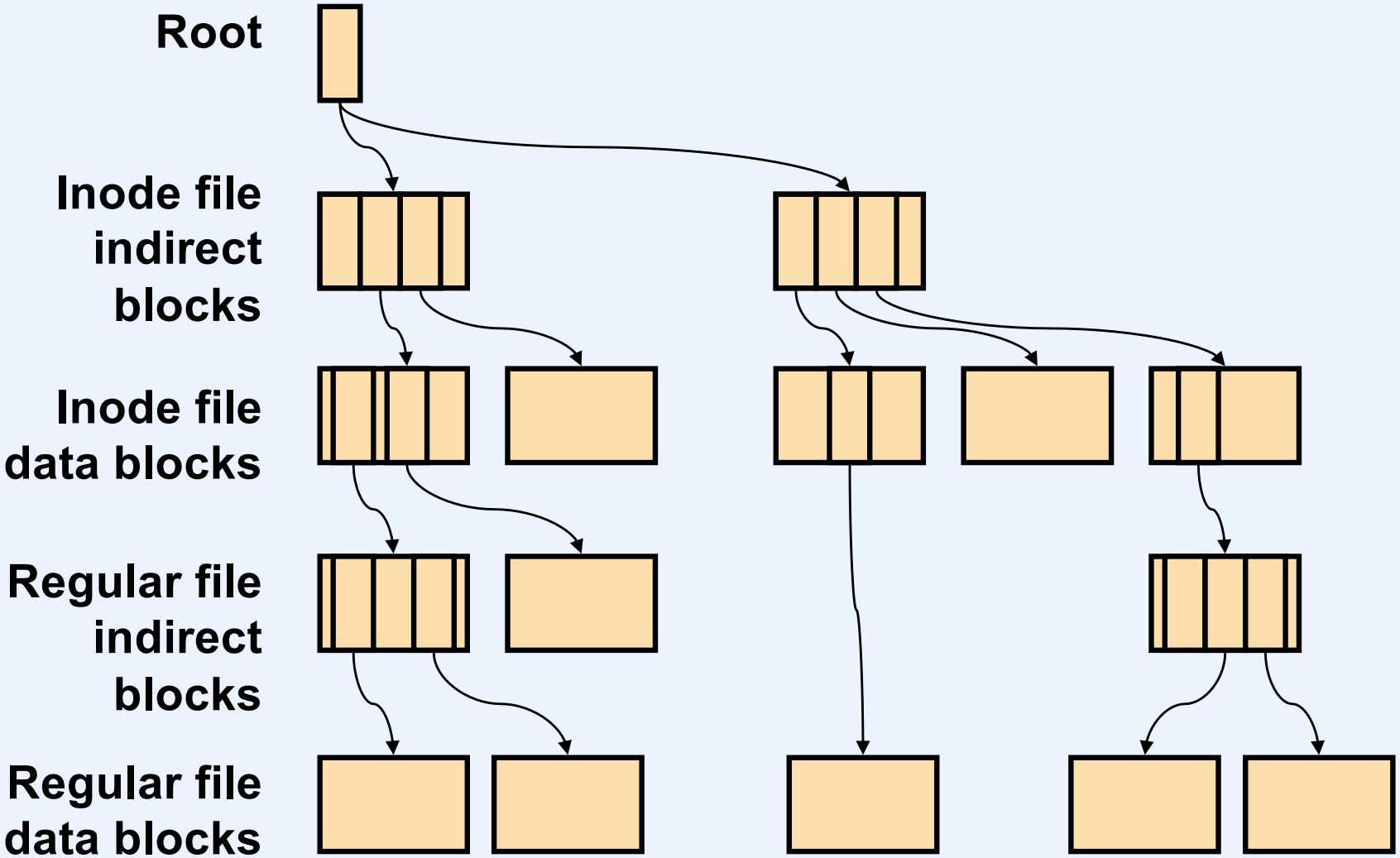
File Systems Part 4

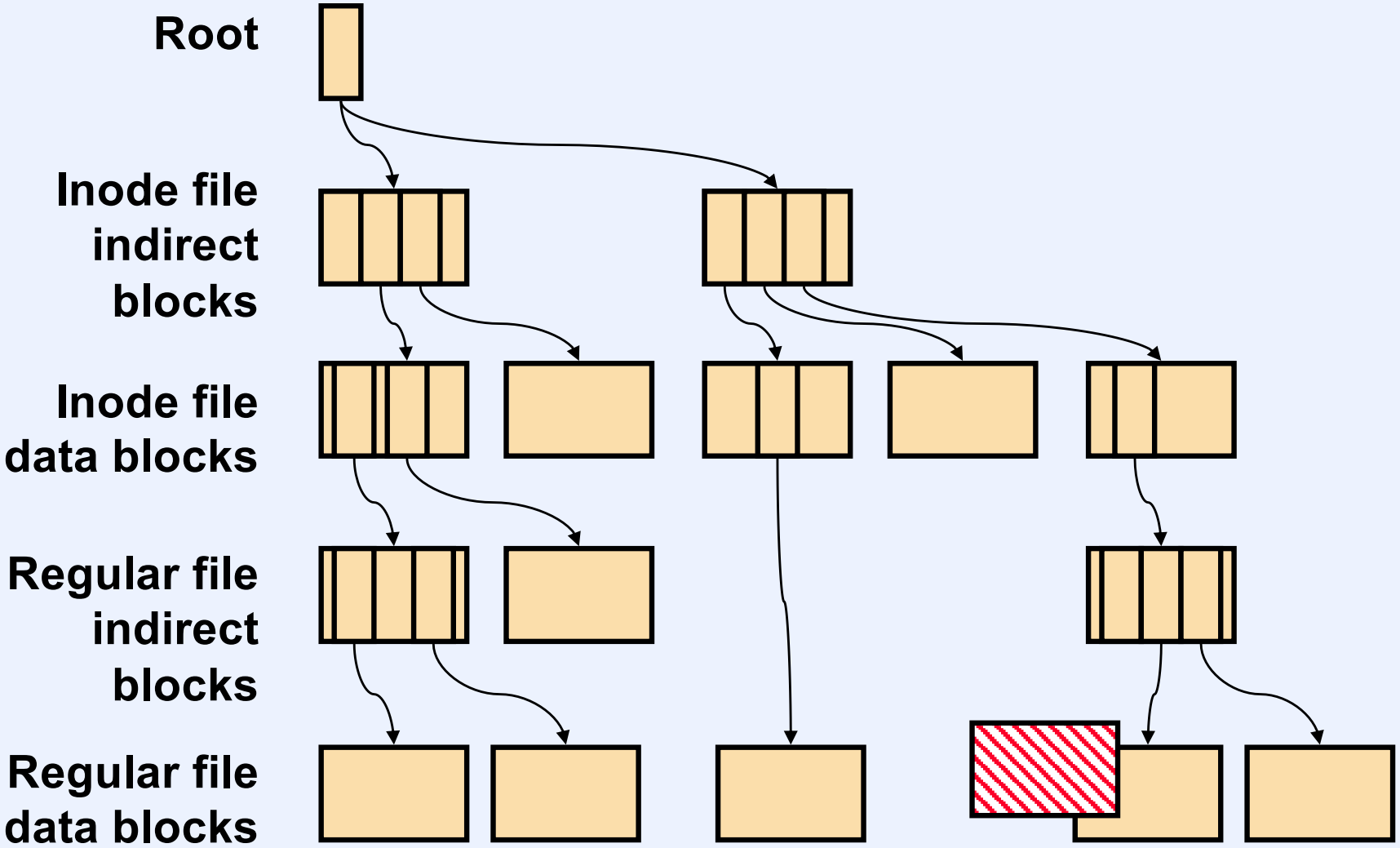
Shadow Paging

- Refreshingly simple
- Provides historical snapshots
- Examples
  - WAFL (Network Appliance)
  - ZFS (Sun)

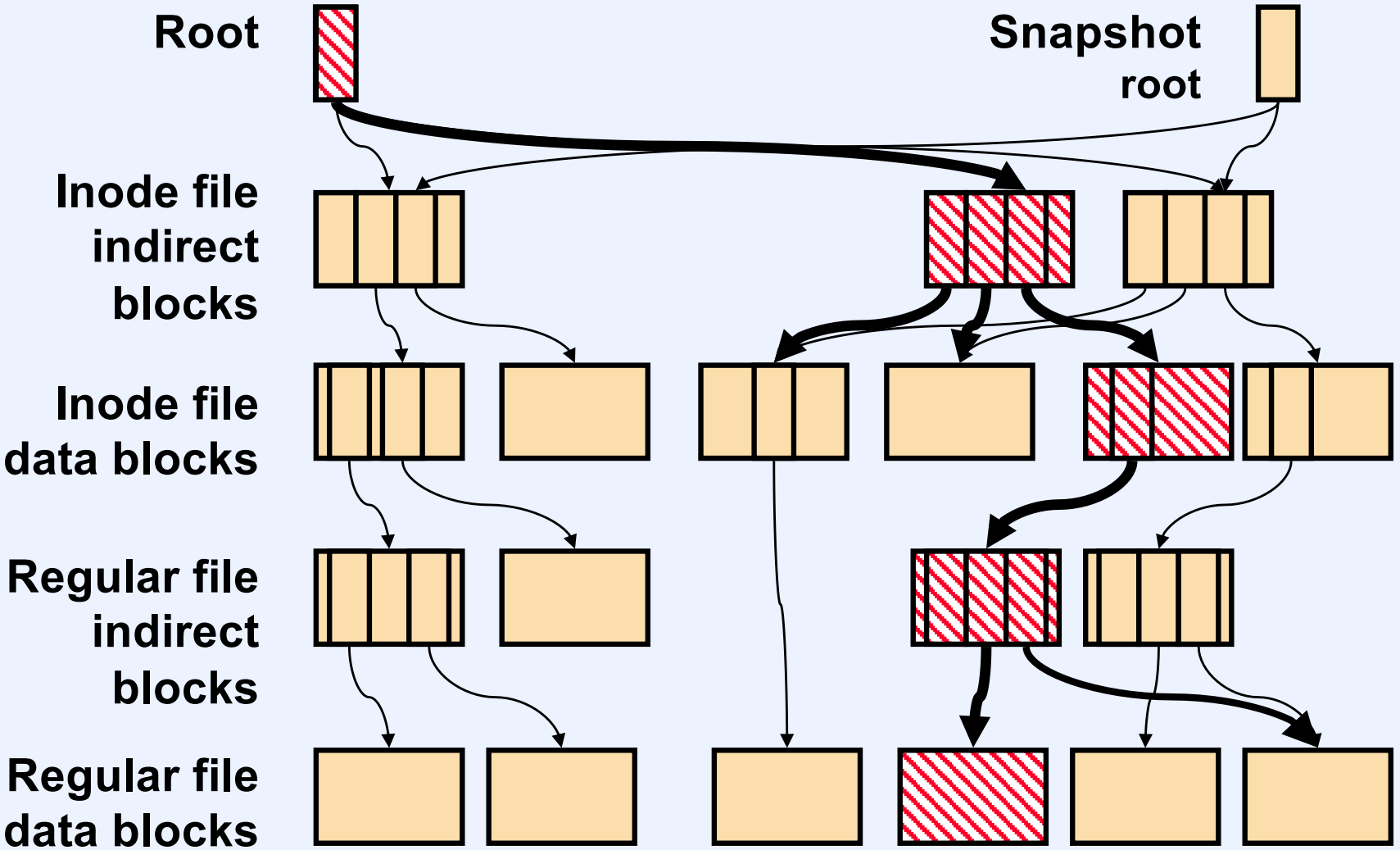
Shadow-Page Tree



Shadow-Page Tree: Modifying a Node



Shadow-Page Tree: Propagating Changes



Benefits

- Each update results in a new shadow-page tree (having much in common with the previous one)
- The current root identifies the current tree
- If the system crashes after an update has been made, but before changes are propagated to the new root, the update is lost
  - a single write (to the root) effectively serves as a commit
- Older roots refer to previous states of the file system – snapshots

Quiz 1

When the shadow-page tree is updated:

a) all file-system data must be written to disk synchronously: writes may be cached for the sake of reads, but write system calls may not return until the data is on disk

b) file-system data may be cached (and written asynchronously) only if all lower parts of the tree are written to disk before upper parts

c) file-system data may be cached (and written asynchronously) as long as the root is written last

Desirable Properties of Directories

- No restrictions on names
- Fast
- Space-efficient

Implementation Strategies

- Fixed-sized entries
- Variable-sized entries

- Sequential search
- Hashing
- Balanced search trees

S5FS Directories

Component Name	Inode Number

directory entry

.	1
..	1
unix	117
etc	4
home	18
pro	36
dev	93

Quiz 2

Deleting an S5FS directory entry entails simply zeroing out the entry. No compaction of entries is done, even if a great number are deleted. Why not?

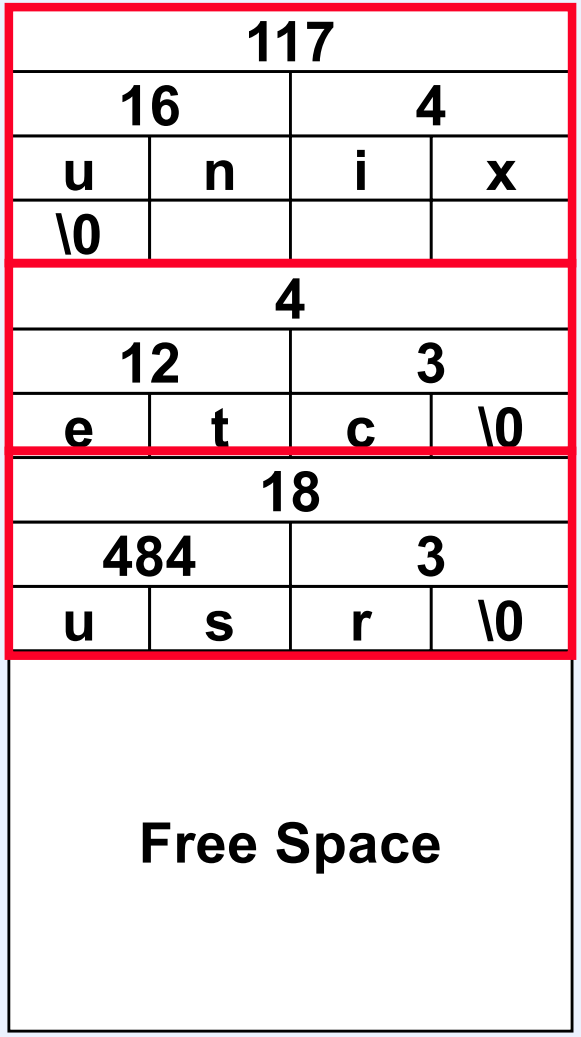
a) The implementer was lazy

b) It's really difficult to do right

c) Deletion is so rare that it's not worth worrying about

d) It never occurred to them to do it

FFS Directory Format



Directory Block

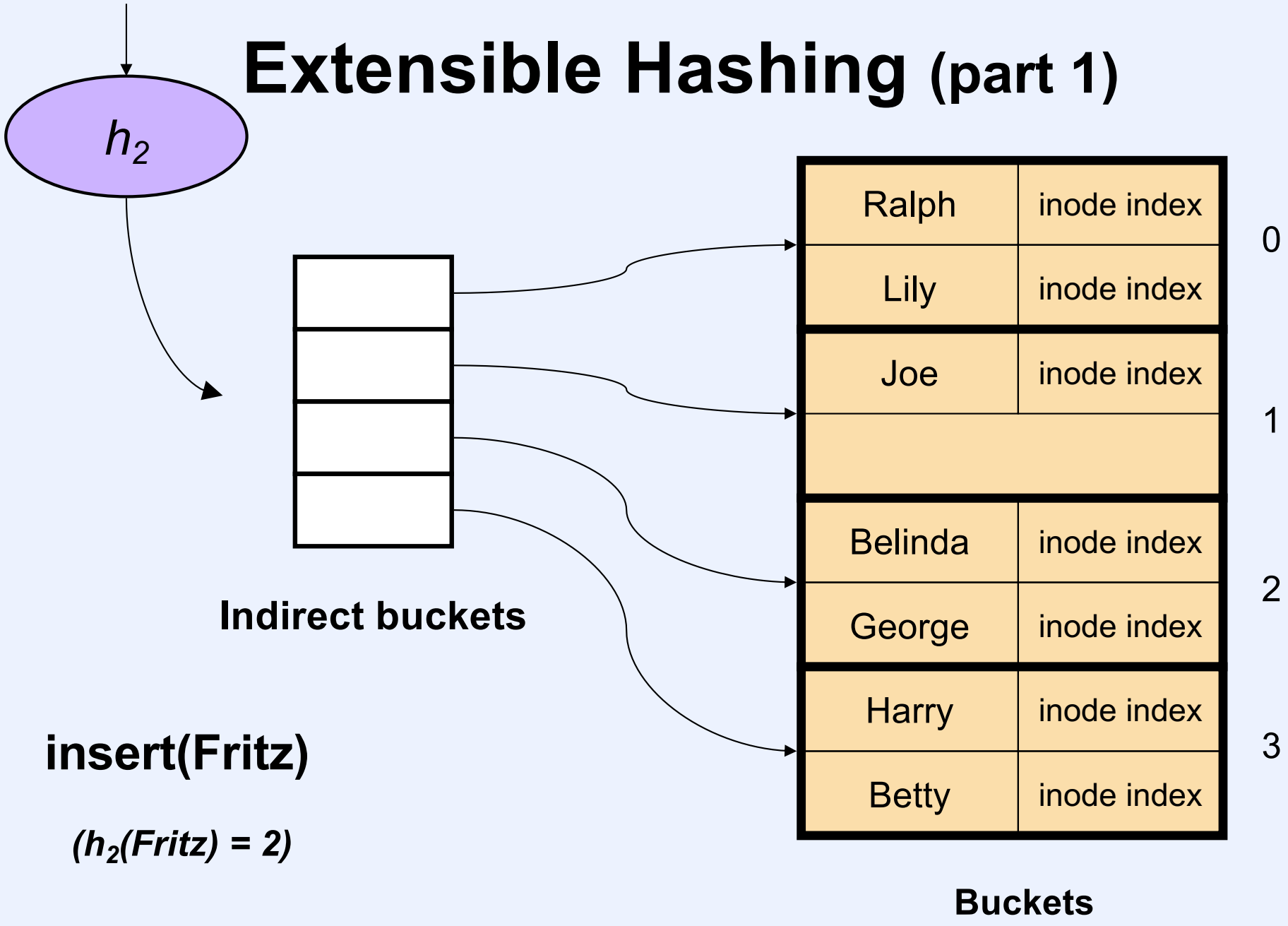
Hashed Directories

- Implement directory using a hash table
  - h = hash function
  - h(name) yields either directory entry or not present
  - how large is the hash table?
    - can it increase in size if directory grows?
  - use *extensible hashing*
    - as directory grows, hash table grows and new hash functions are employed

Extensible Hash Functions

- Sequence of functions h_0, h_1, h_2, ...
  - h_i maps strings into 2^i indirect buckets
  - i low-order bits of h_{i+1}(component) are the same as the i bits of h_i(component)
    - h_2("Adam") is 01
    - h_3("Adam") is 001 or 101

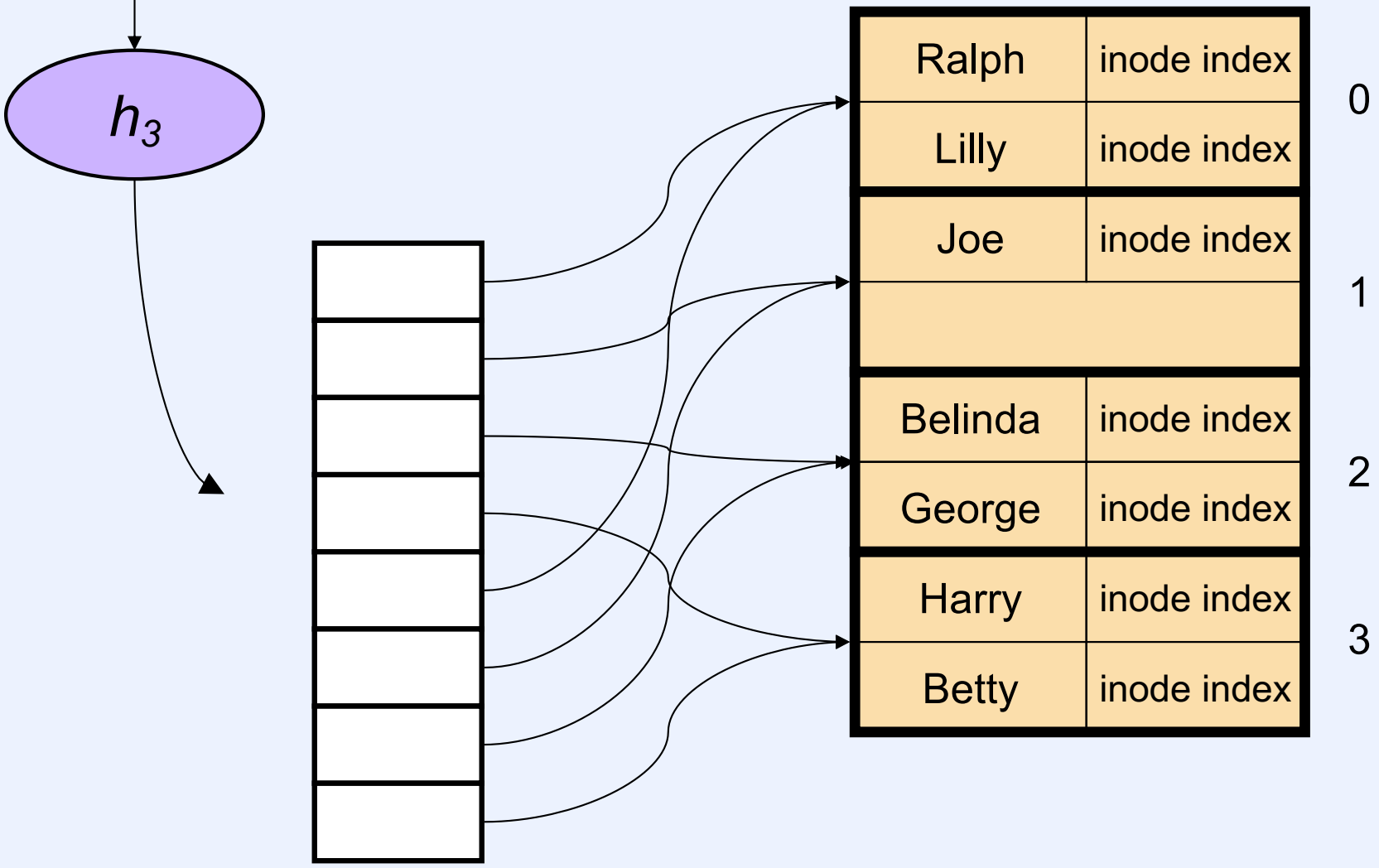
Extensible Hashing (part 1)



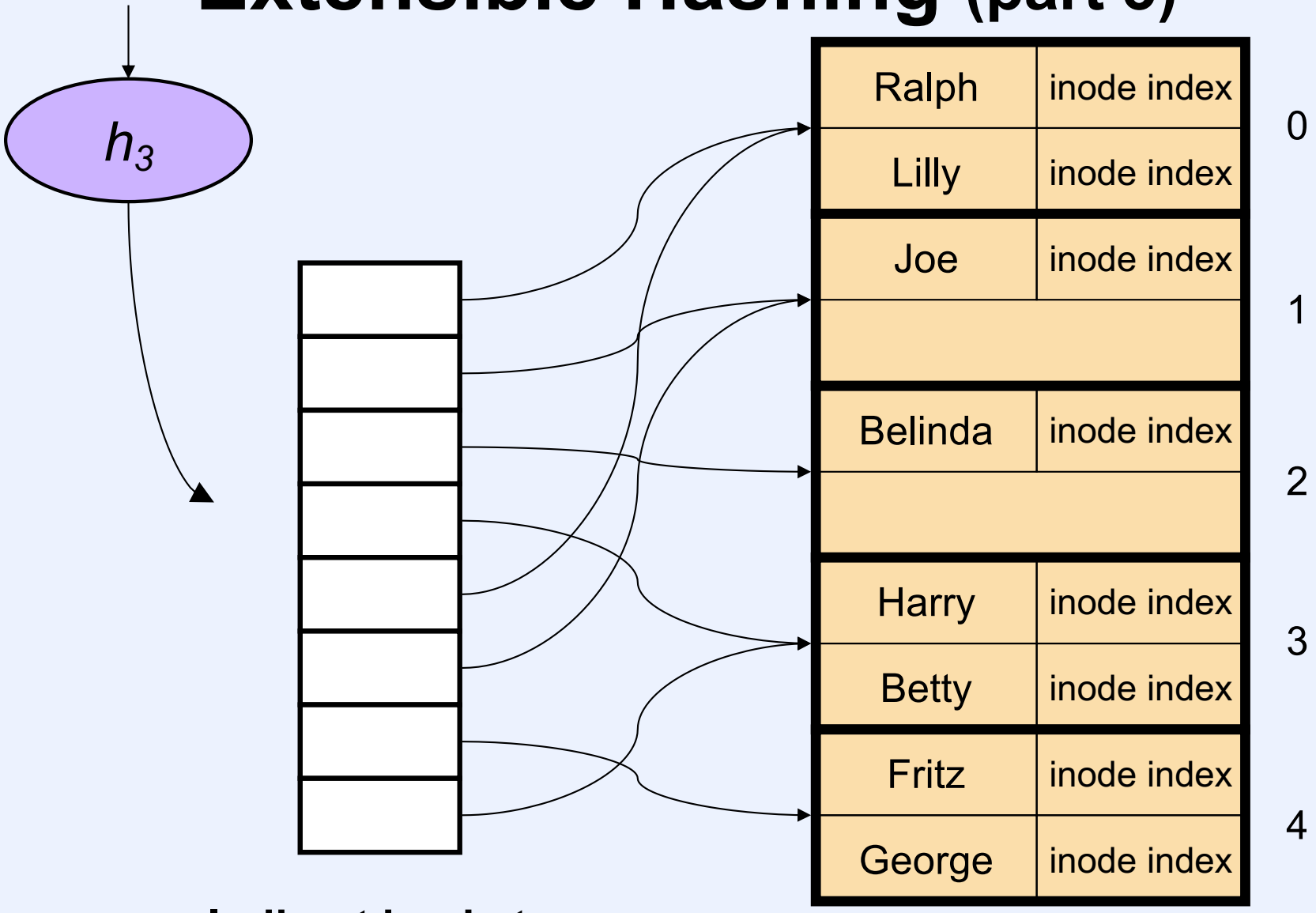
insert(Fritz)

(h_2(Fritz) = 2)

Extensible Hashing (part 2)



Extensible Hashing (part 3)



Quiz 3

The number of disk reads required to search for an entry in an S5FS directory of size n is O(n).

How many disk reads are required to search for an entry in a directory implemented using extensible hashing? Assume the directory is contained within a file.

a) O(1)

b) O(log n)

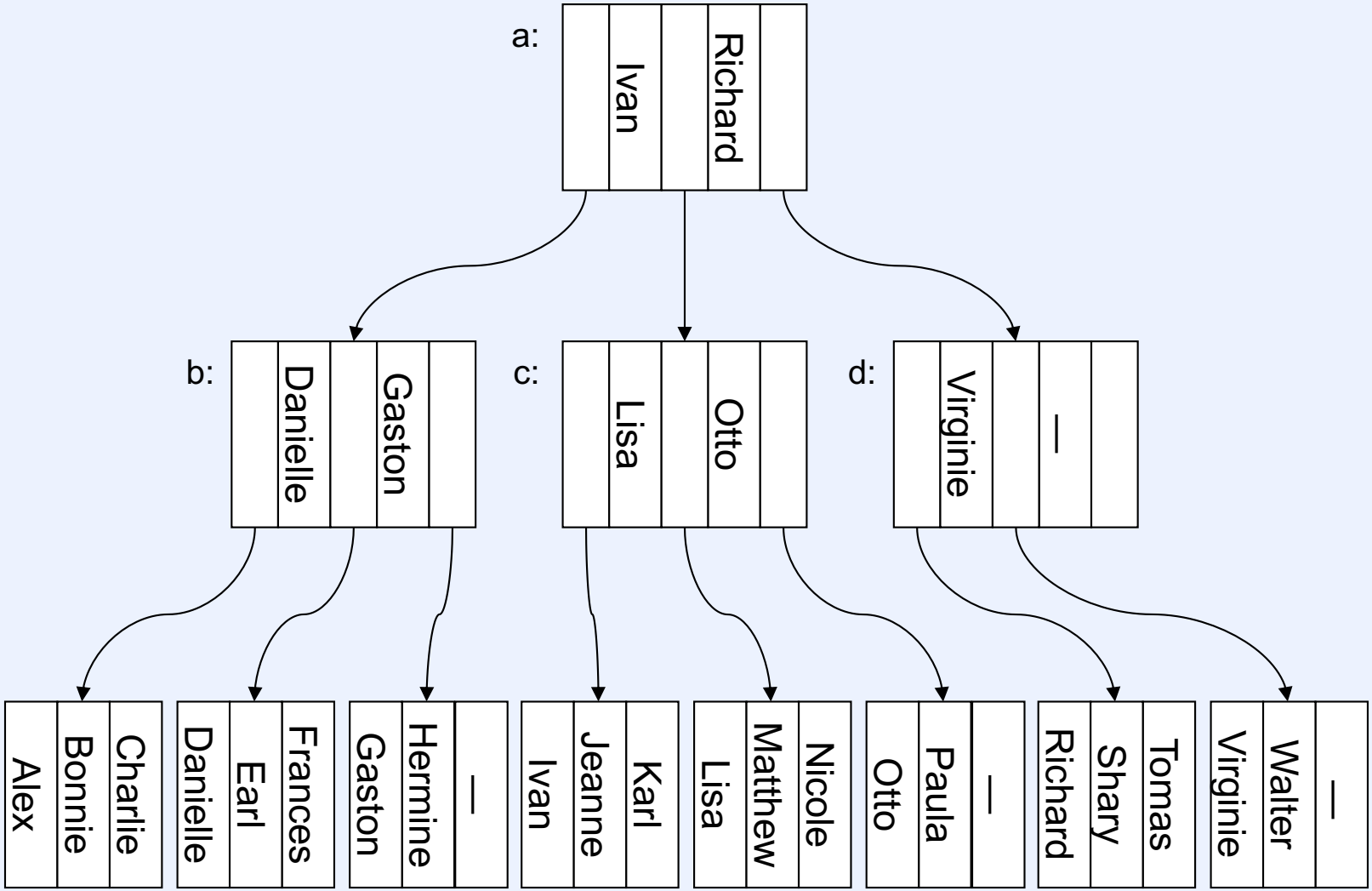
c) O(n)

d) O(n·log n)

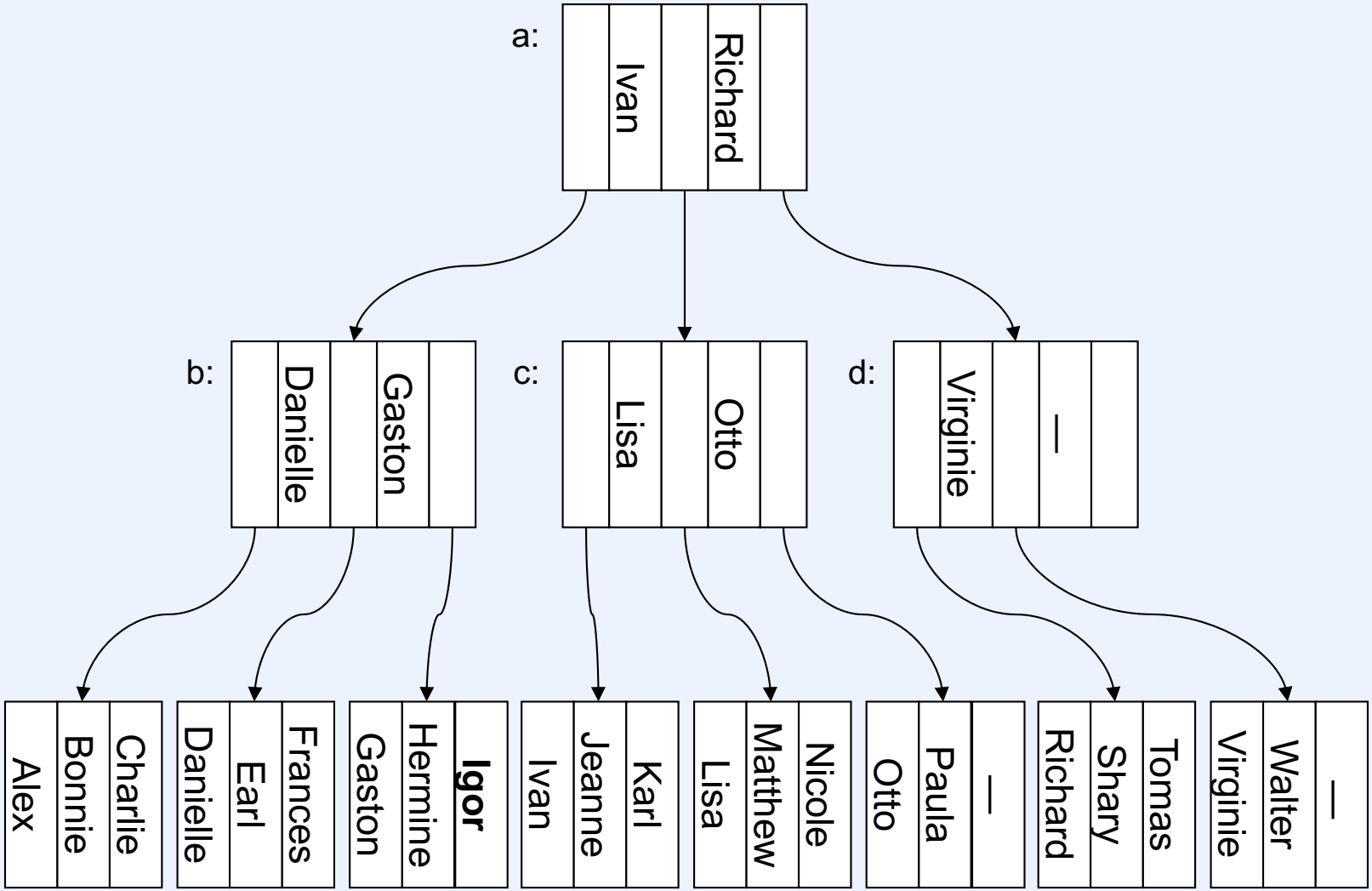
Balanced Search Trees

- Implement directory as a balanced search tree
  - easily grown to accommodate larger directories, but still log n searches
  - a common operation is to list the directory in alphabetical order
    - should be cheap
  - use B+ trees

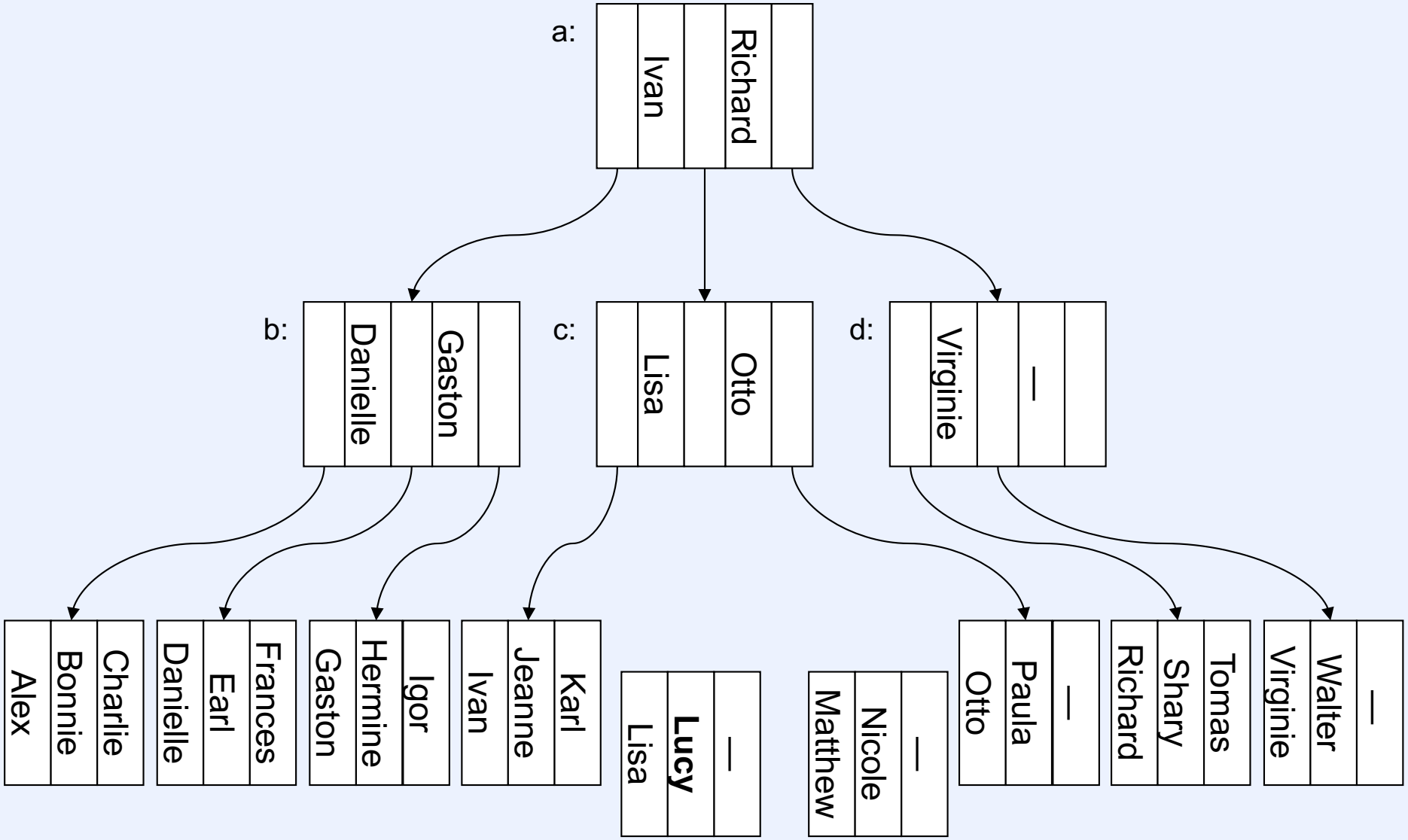
B+ Trees (part 1)



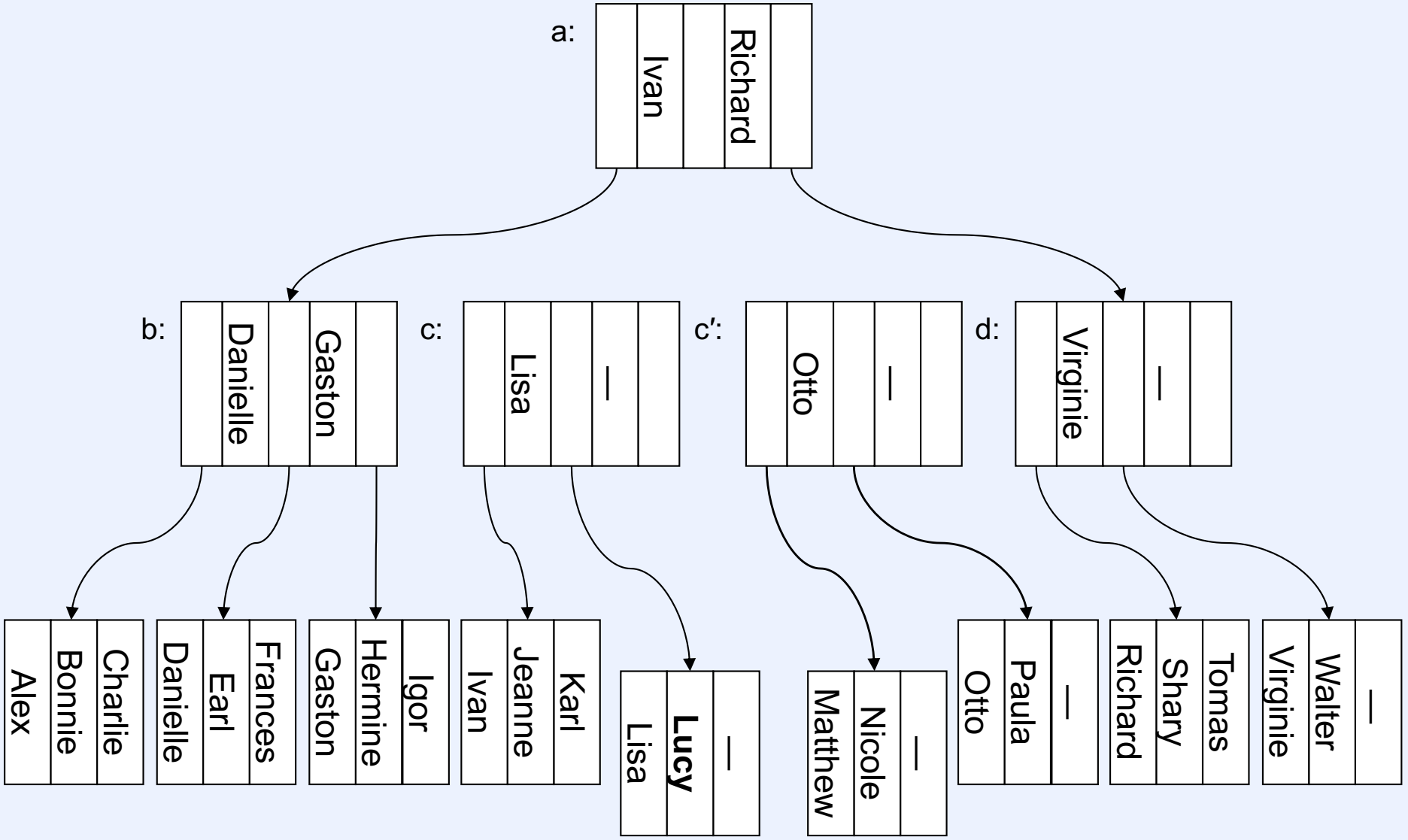
B+ Trees (part 2)



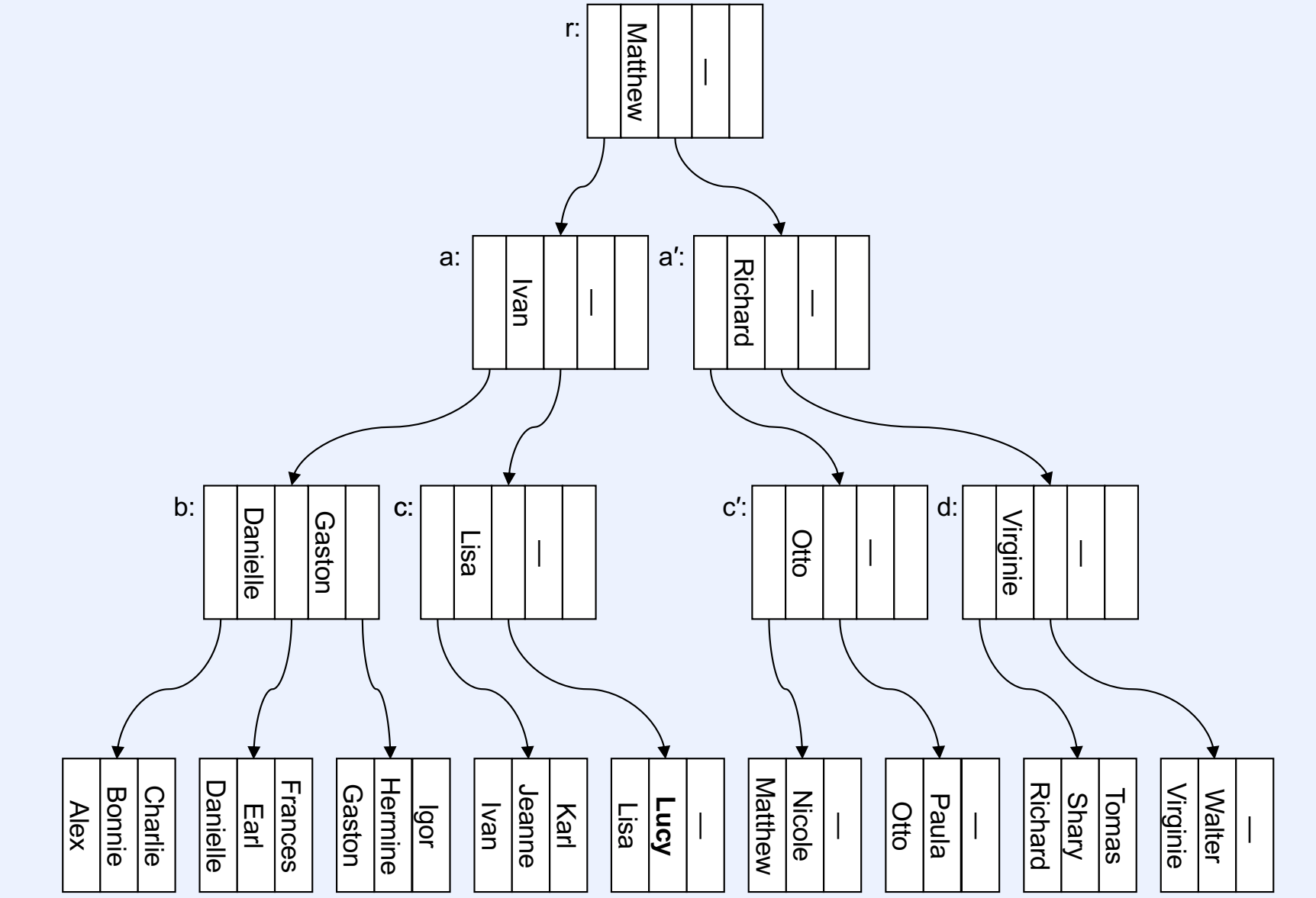
B+ Trees (part 3)



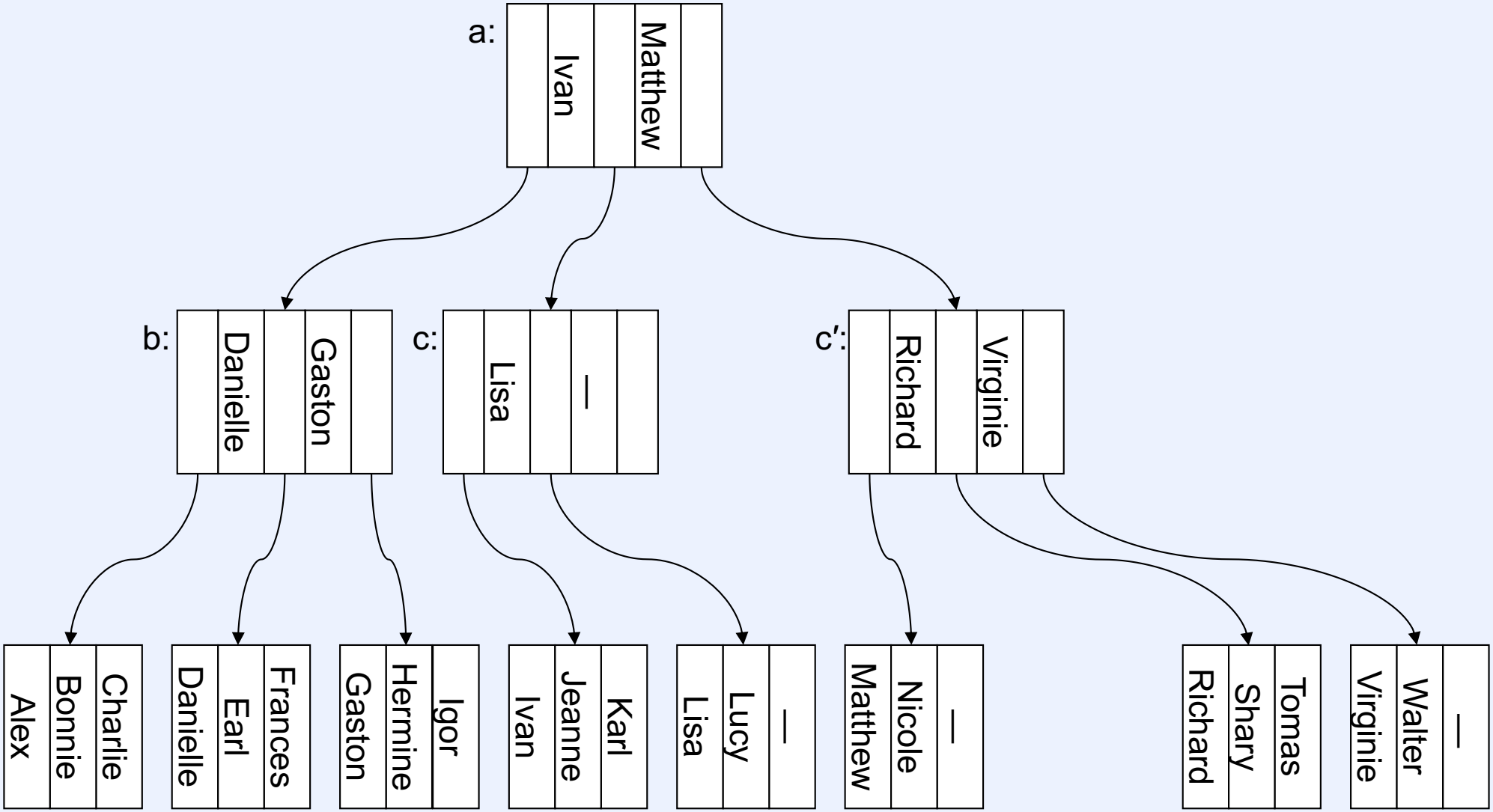
B+ Trees (part 4)



B+ Trees (part 5)



B+ Trees (part 6)



Quiz 4

For what size directories does the B+ tree approach use fewer disk accesses for a lookup than does the extensible hashing approach? Assume each B+ tree node is in a separate block.

a) really small directories

b) really large directories

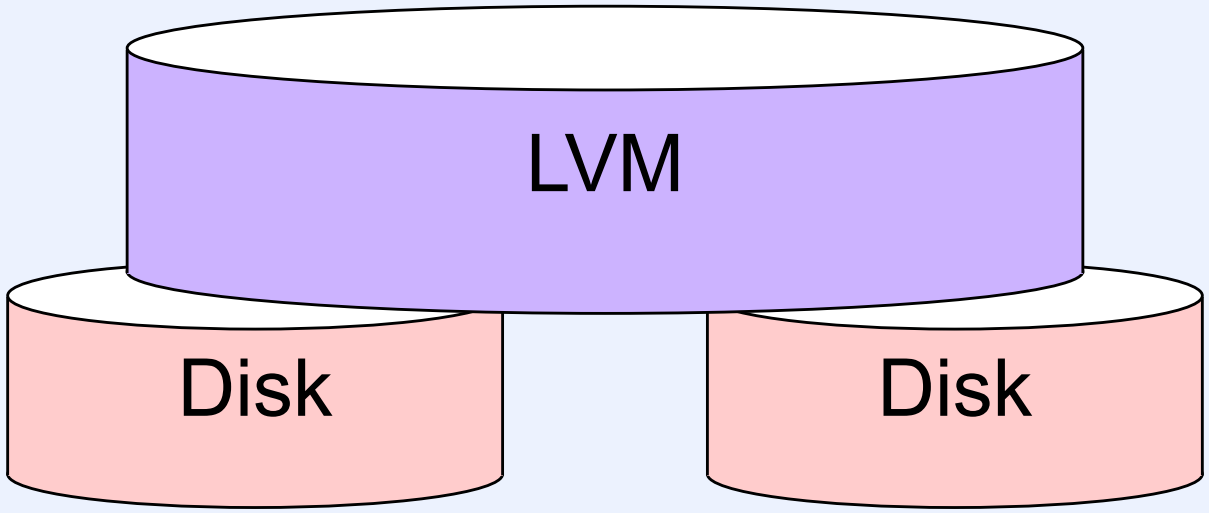
c) it always uses fewer disk accesses

d) it never uses fewer disk accesses

Benefits of Multiple Disks

- They hold more data than one disk does
- Data can be stored redundantly so that if one disk fails, they can be found on another
- Data can be spread across multiple drives, allowing parallel access

Logical Volume Manager



- **Spanning**
  - **two real disks appear to file system as one large disk**
- **Mirroring**
  - **file system writes redundantly to both disks**
  - **reads from one**

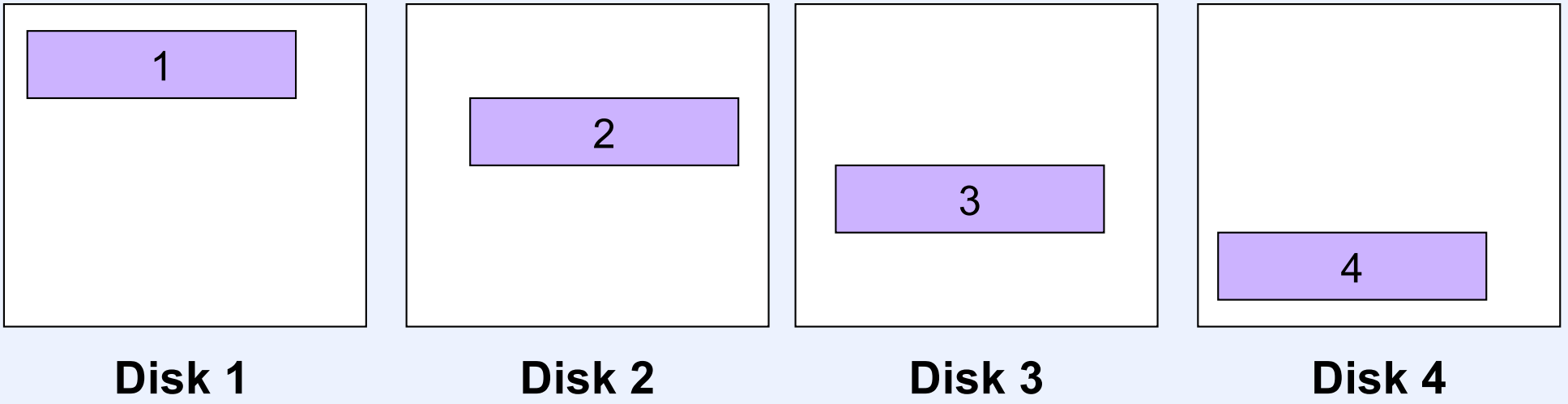
Striping

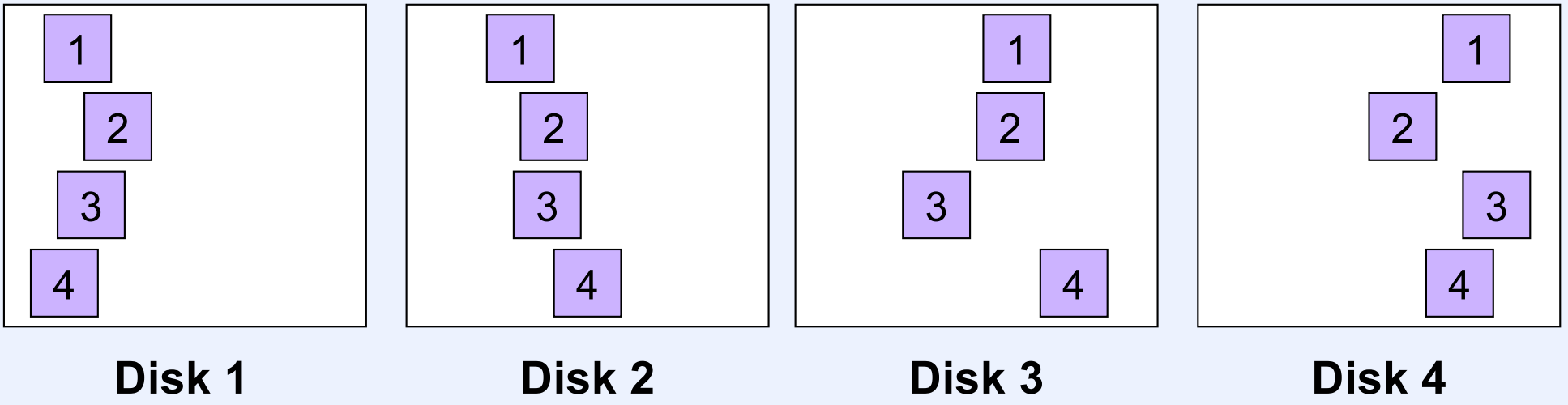
	Disk 1	Disk 2	Disk 3	Disk 4
Stripe 1	Unit 1	Unit 2	Unit 3	Unit 4
Stripe 2	Unit 5	Unit 6	Unit 7	Unit 8
Stripe 3	Unit 9	Unit 10	Unit 11	Unit 12
Stripe 4	Unit 13	Unit 14	Unit 15	Unit 16
Stripe 5	Unit 17	Unit 18	Unit 19	Unit 20

Concurrency Factor

- How many requests are available to be executed at once?
  - one request in queue at a time
    - concurrency factor = 1
    - e.g., one single-threaded application placing one request at a time
  - many requests in queue
    - concurrency factor > 1
    - e.g., multiple threads placing file-system requests

Striping Unit Size





Quiz 5

In the previous slide, suppose we have four threads concurrently reading from the disks. The first is reading all the sectors labeled one, the second is reading all the sectors labeled two, the third three, and the fourth four. With which layout would they complete all the transfers the quickest?

a) top

b) bottom

c) roughly equal

Quiz 6

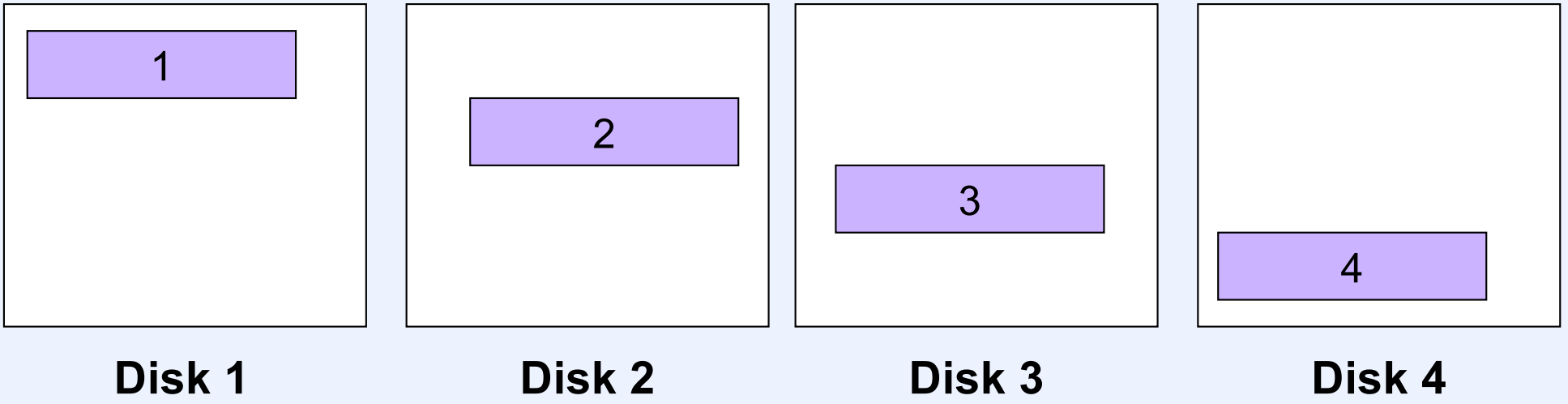
Now suppose we have one thread that first reads the sectors labeled one, then those labeled two, then three, then four. With which layout would it complete the transfers the quickest?

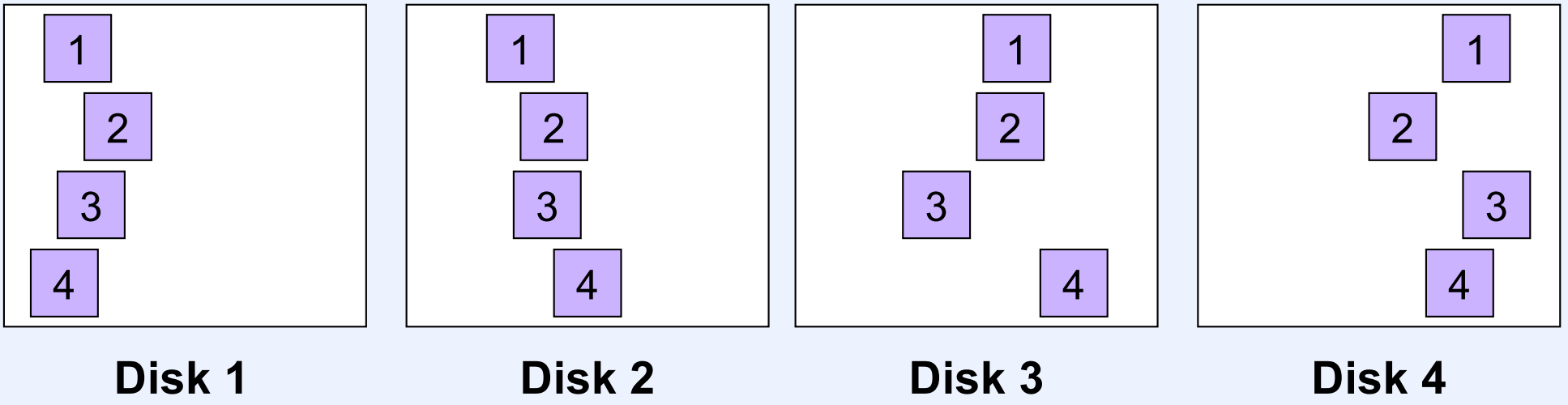
a) top

b) bottom

c) roughly equal

Striping Unit Size (Again)





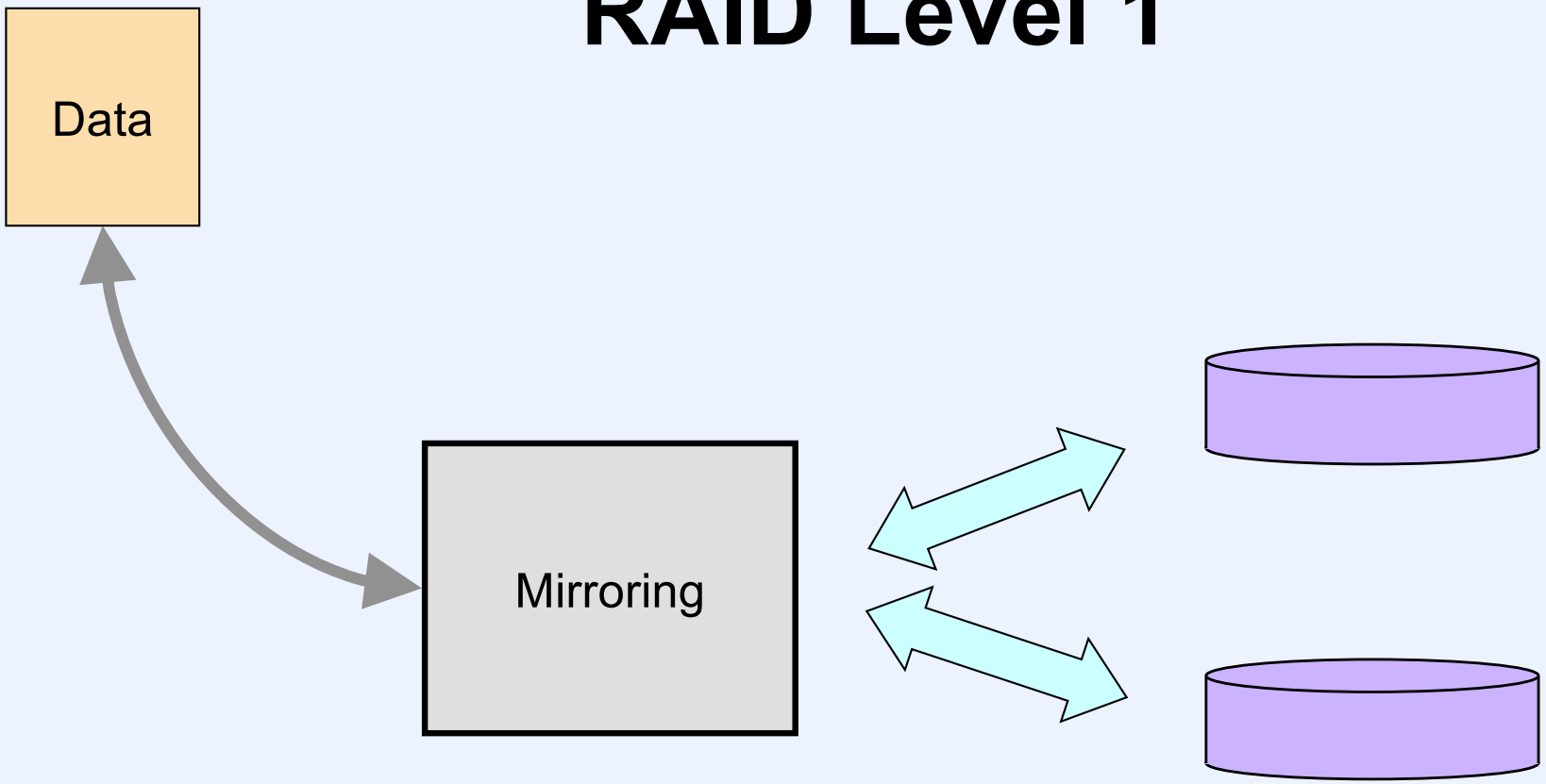
Striping: The Effective Disk

- Improved effective transfer speed
  - parallelism
- No improvement in seek and rotational delays
  - sometimes worse
- A system depending on N disks is much more likely to fail than one depending on one disk
  - if probability of one disk's failing is *f*
  - probability of N-disk system's failing is $(1-(1-f)^N)$
  - (assumes failures are IID, which is probably wrong …)

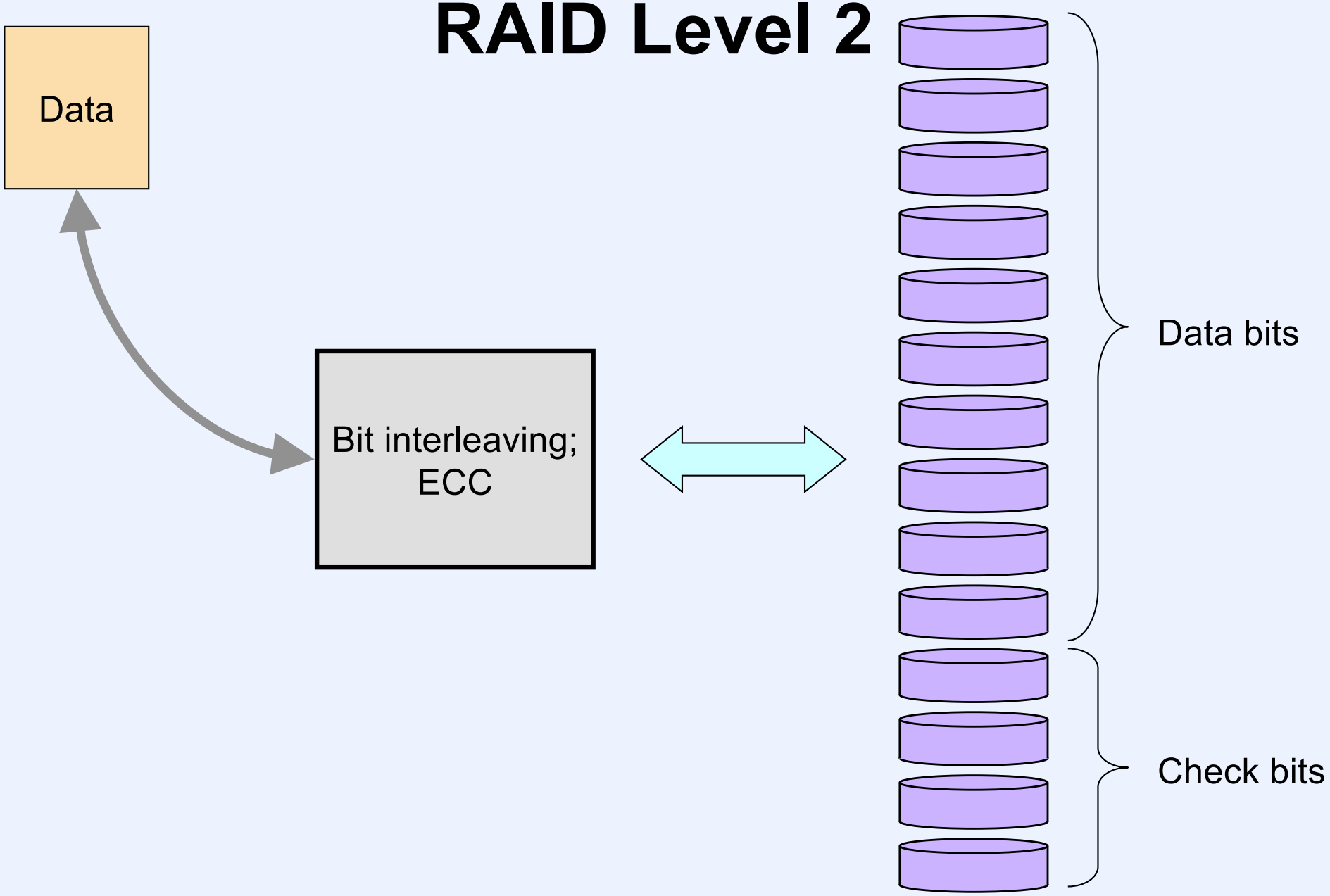
RAID to the Rescue

- Redundant Array of Inexpensive Disks
  - (as opposed to Single Large Expensive Disk: SLED)
  - combine striping with mirroring
  - 5 different variations originally defined
    - RAID level 1 through RAID level 5
      - RAID level 0: pure striping
        - numbering extended later
      - RAID level 1: pure mirroring

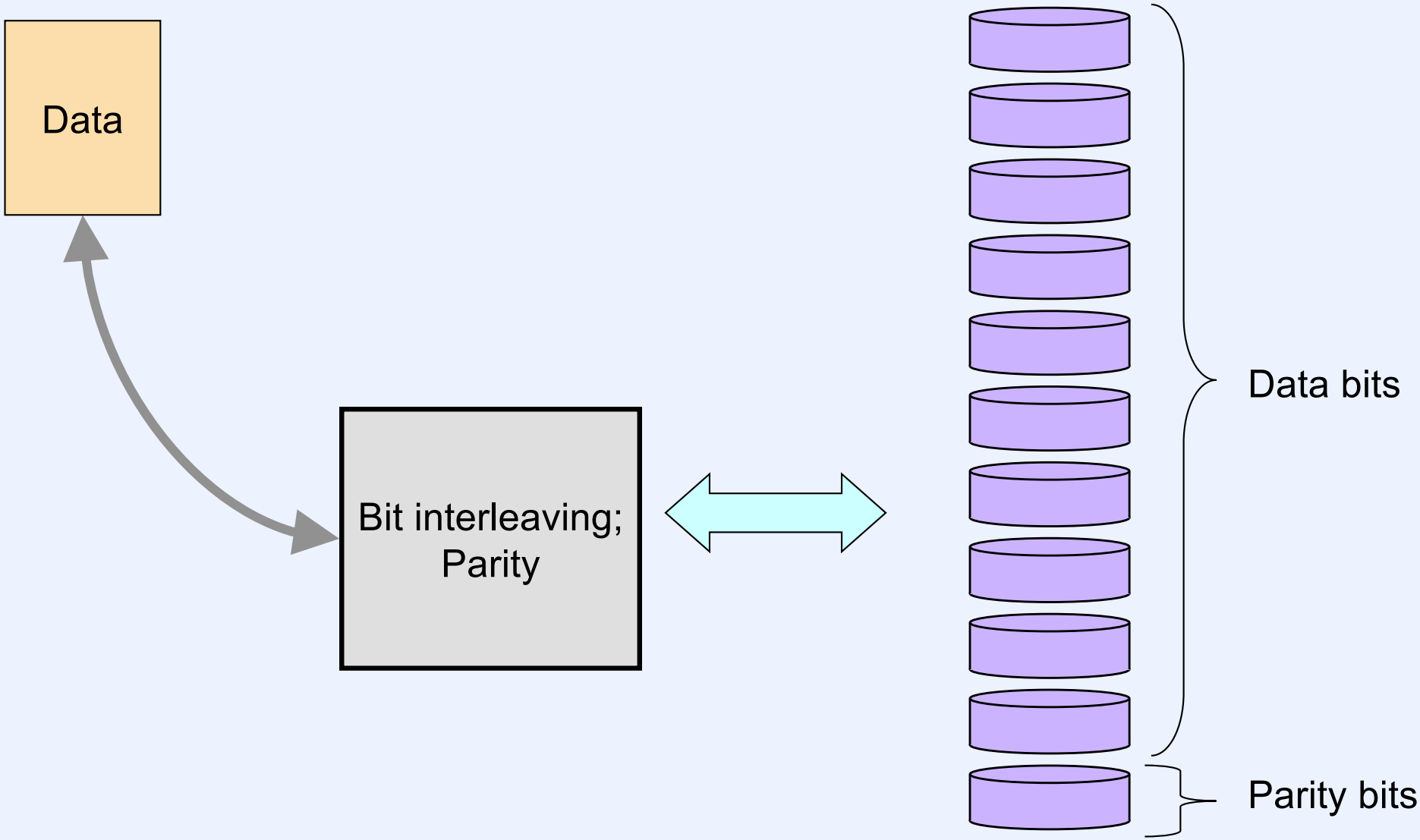
RAID Level 1



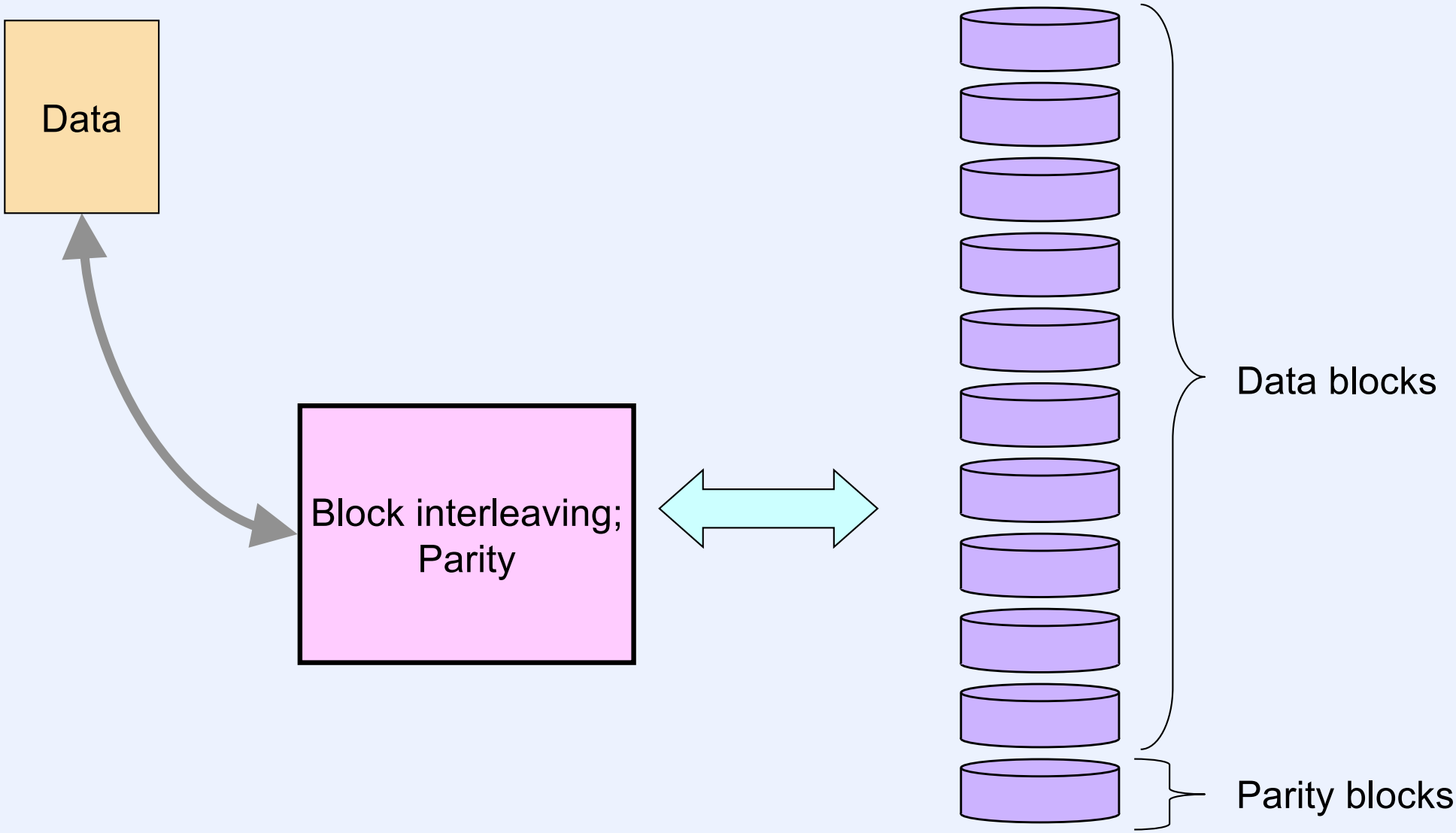
RAID Level 2



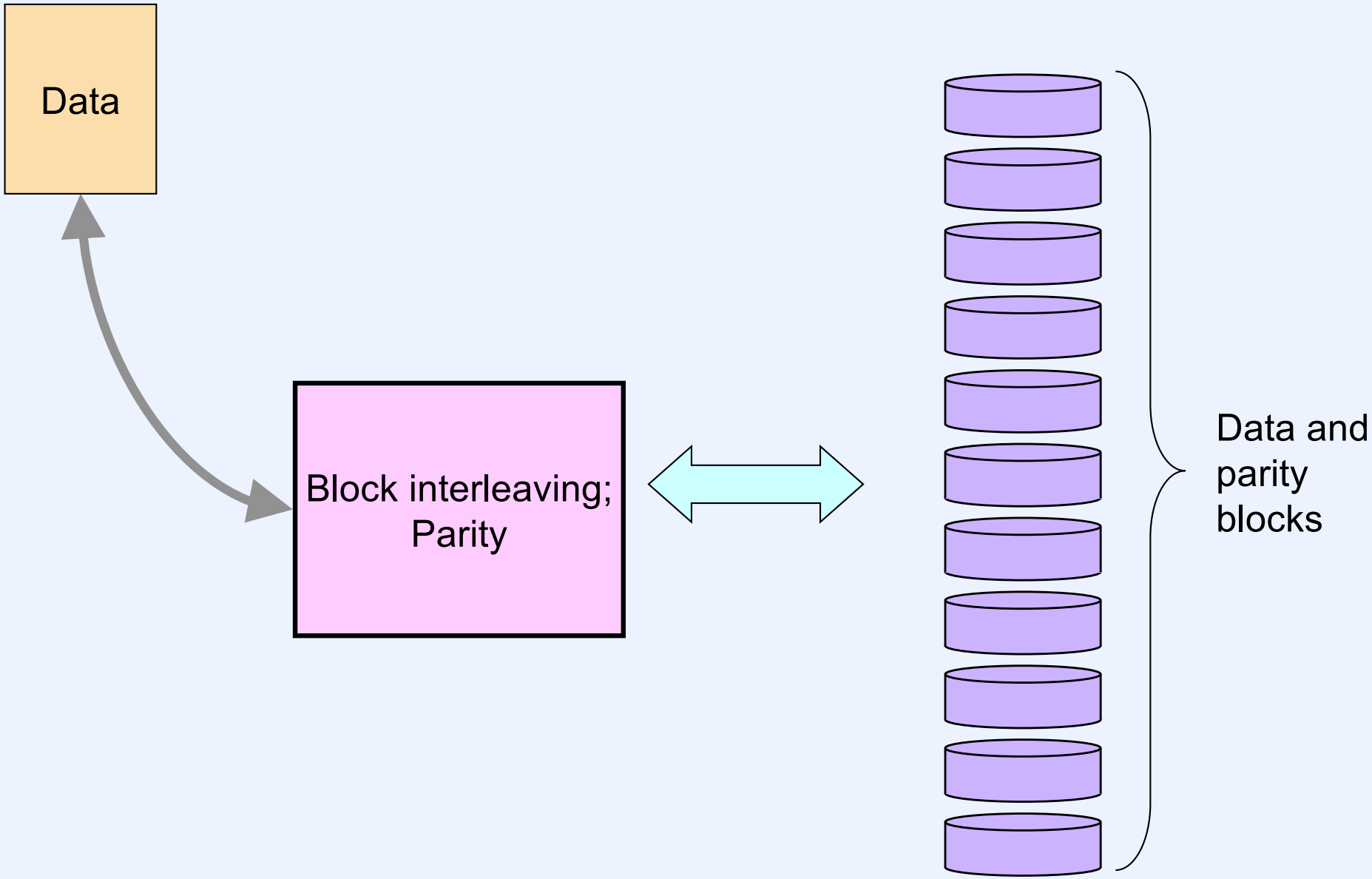
RAID Level 3



RAID Level 4



RAID Level 5



RAID 4 vs. RAID 5

- Lots of small writes
  - RAID 5 is best
- Mostly large writes
  - multiples of stripes
  - either is fine