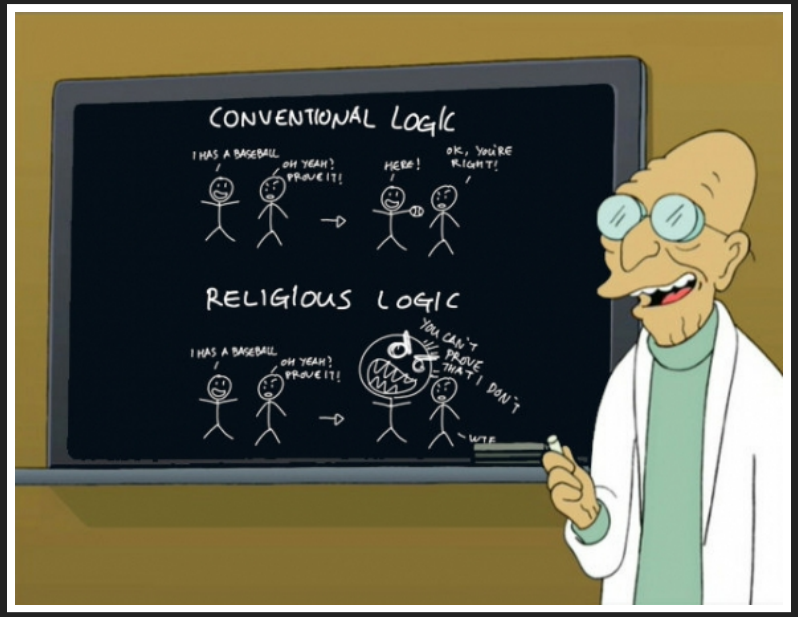
DISTRIBUTED STORAGE FOR CLOUD APPLICATIONS

A (very fast) overview

presented by Dimitri Pertin @ Polytech Nantes - 20/11/15

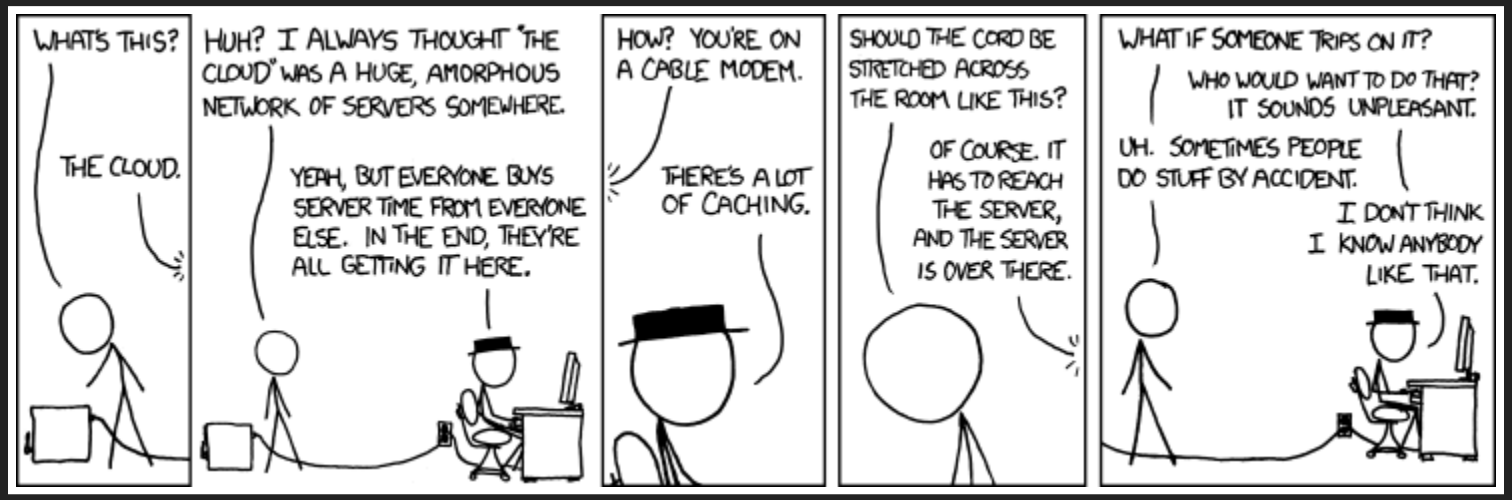
WHOAMI



"Geometrical approach to design distributed-storage-oriented erasure codes"

CLOUD STORAGE

"There is no cloud. It's just someone else's computer"



from xkcd

CLOUD APPLICATIONS

MULTIMEDIA


BRUTAL LEGEND
DiRT3
GRID 2
GRID
F.E.A.R.
ALAN WAKE
DEAD RISING
THE WALKING DEAD
SAINTS ROW

BIG DATA


BIG
DATA
INFORMATION
STORAGE
LARGE
SETS
PETABYTES
SEARCH
ANALYSIS
RESEARCH
MANAGEMENT
TECHNOLOGIES
SOFTWARE
MILLION
VOLUME
GOVERNMENT
DEVELOPMENT
TECHNOLOGY
SCIENCE
FUTURE
NEW
INTERNET
LEARNING
EVERY WORLD
BASED
PROCESS
SAN
NAS
SOCIAL
YEARS
RESULTS
DRIVE
REPLICATION
HANDLE
SOURCES
CHALLENGES
LIMITS
COMPLEX
ANNOUNCED
PAST
HAND
CAPACITY
ONE
FACTOR
SYSTEMS
MANAGE
ONLINE
DAY
WORLDWIDE
MAKING
INTEGRATION
MASSIVE
DEFINITION
USING
FRAMEWORK
BETWEEN
STATISTICS
BUSINESS
EFFECTIVE
VISUALIZATION
DATABASES
SIZE
ANALYTICS
SIMULATIONS
PROCESSING
EXABYTES
DETERMINE
WELL
SET
DEPARTMENT
QUERIES
HIGH
USERS
HIGHER
ALGORITHMS
COMPANIES
AMOUNT
RATE
REAL
STRUCTURE
WORK
APPLICATIONS
INFRASTRUCTURE
TERMS
YEAR
PRIVATE
CAPABILITIES
SUCCESSFUL
NEEDED
USES
LESS
HUGE
COMPARED
VARIOUS
TIMES
SHARED
PRIVACY
INTERNATIONAL
IMPLEMENTATION
BECOMES
SECOND
NATIONAL
INTELLIGENCE
APPROACHES
PROCESSES
FLOW
ARCHITECTURE
OUT
INCLUDE
NETWORKS
HYPOTHESIS
END
MOST
NEARLY
MAPREDUCE
CRITIQUES
BILLION
APPROACH
MANY
PARALLEL
STATE
CRITIQUE
USE
PEOPLE
TRAFFIC
CURRENTLY
INITIATIVE
ECONOMIC
TERABYTES
MARKET
VARIETY
RELATED
TIME
SENSOR
WOULD
PROJECT
DISTRIBUTED
RELEVANT
ANNUAL
NOW
USED
PER
RECORDS
COST
PARADIGM
FUNDING
INSIGHT
TOOLS
CENTER

HIGH-PERFORMANCE COMPUTING

LONG-TERM STORAGE

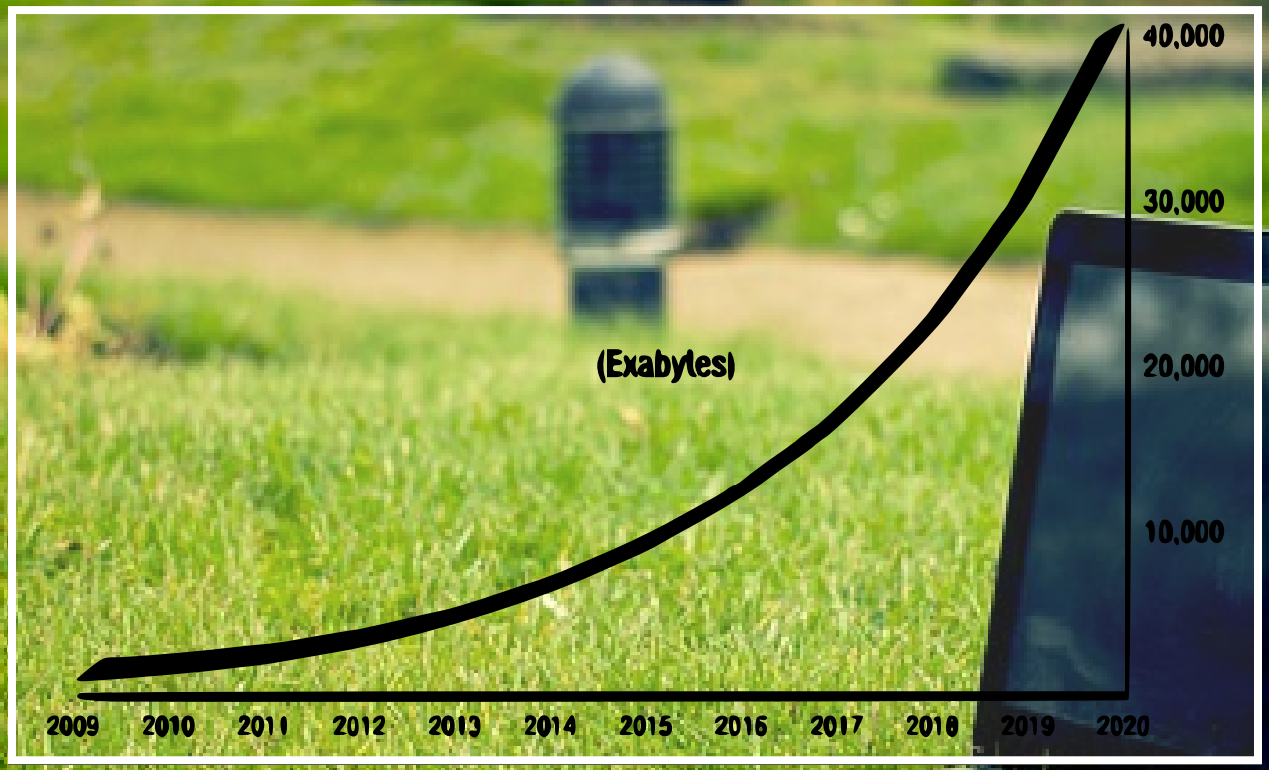
PROBLEMS


RESOURCE ALLOCATION


DIFFERENT TYPES OF HARDWARE
SAMSUNG

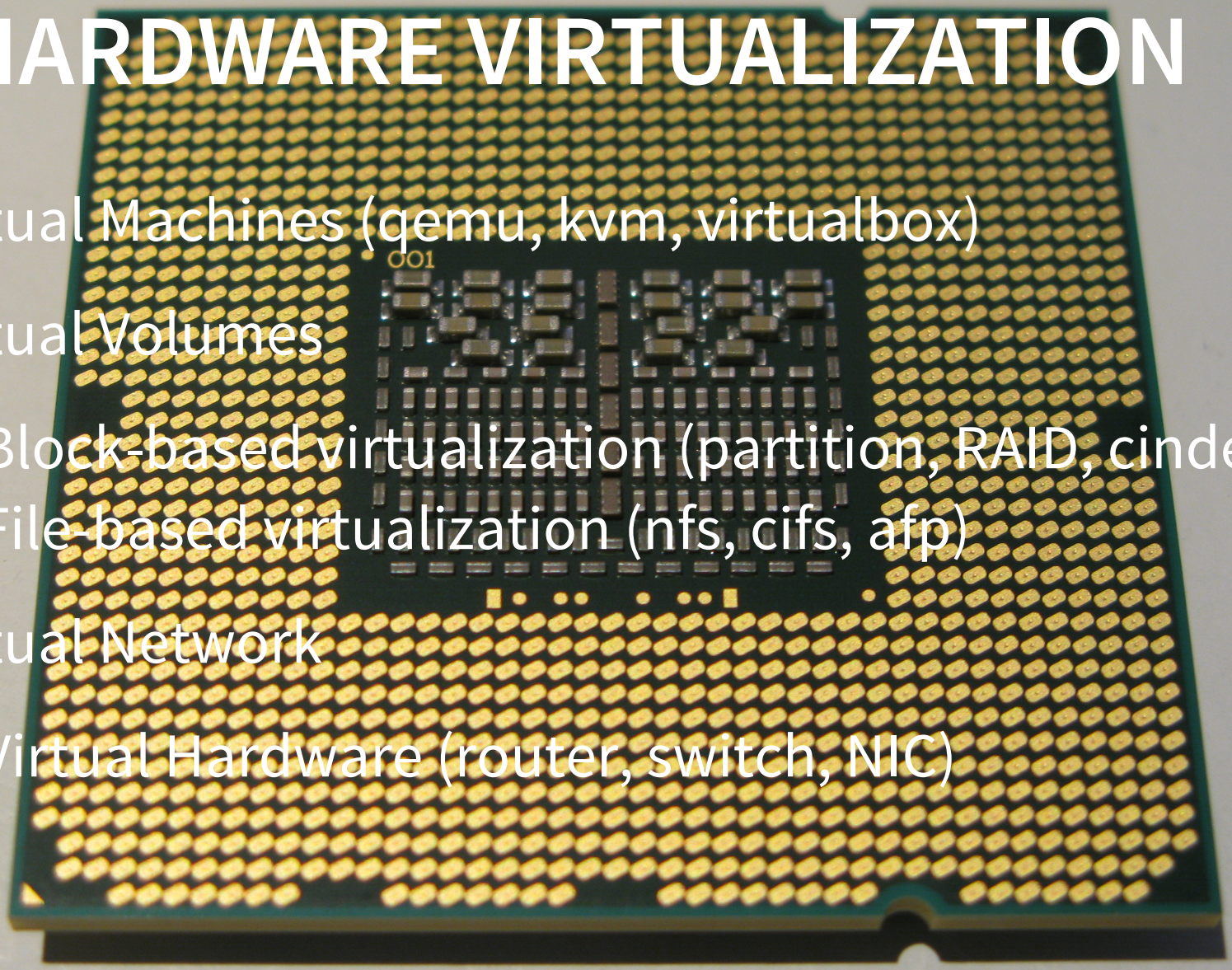
MULTIPLE SOURCES OF FAILURE

COST / ENERGY EFFECTIVE ($$$)



HOLY GRAIL: VIRTUALIZATION

HARDWARE VIRTUALIZATION



- Virtual Machines (qemu, kvm, virtualbox)
- Virtual Volumes
  - Block-based virtualization (partition, RAID, cinder)
  - File-based virtualization (nfs, cifs, afp)
- Virtual Network
  - Virtual Hardware (router, switch, NIC)

SOFTWARE-DEFINED VIRTUALIZATION

- Software-Defined Storage (SDS)
  - File Systems (rozofs, cephfs, hdfs)
- Software-Defined Network (SDN)
  - Virtual elements (firewall, load balancer)
  - OpenFlow (protocol to access forwarding planes)
  - Virtual LAN

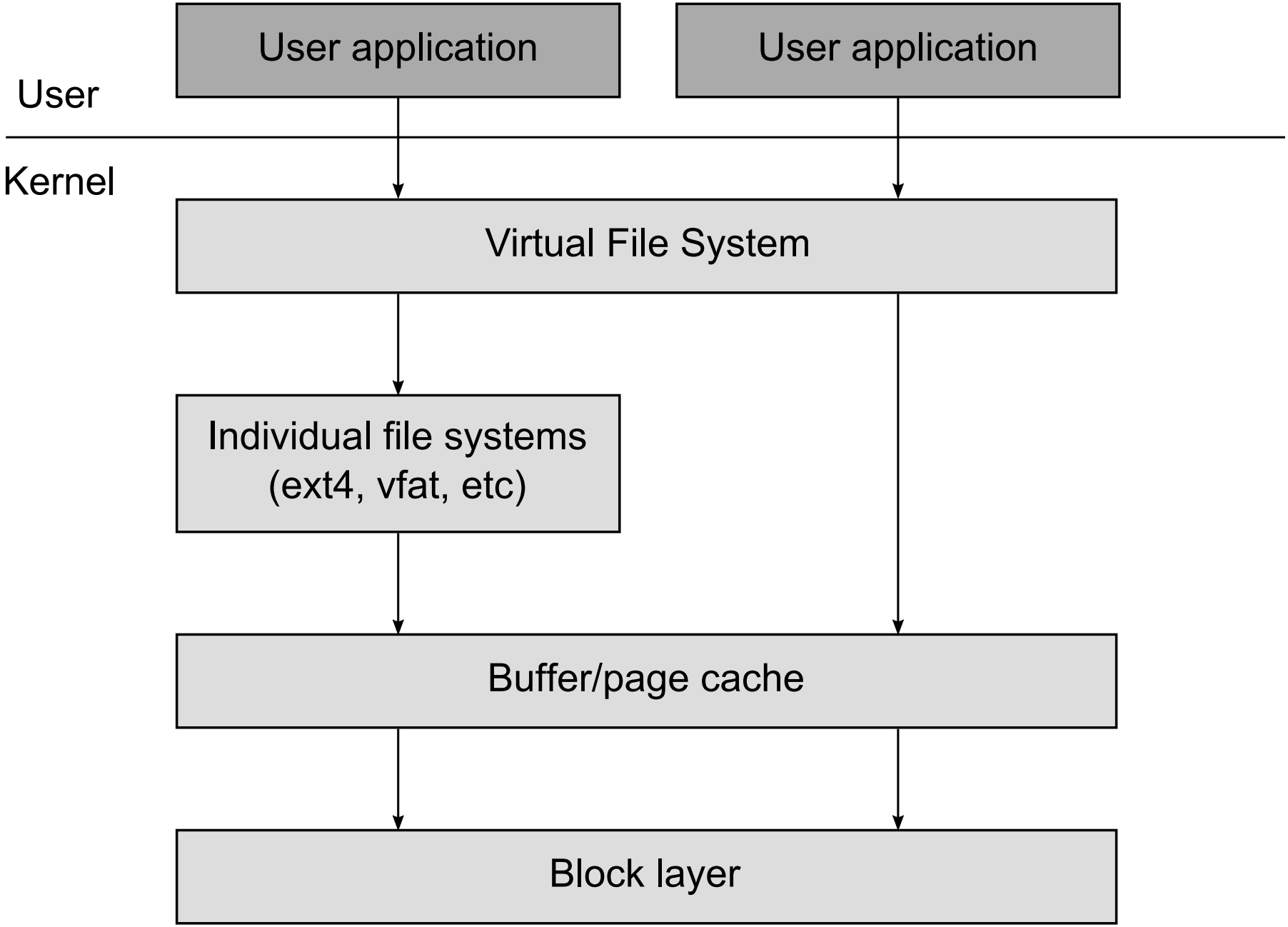
OUTLINE

1. File System Basics
2. Scale-out Network Attached Storage (NAS)
3. Example with RozoFS: a Distributed File System
4. My storage in the Cloud (devops)

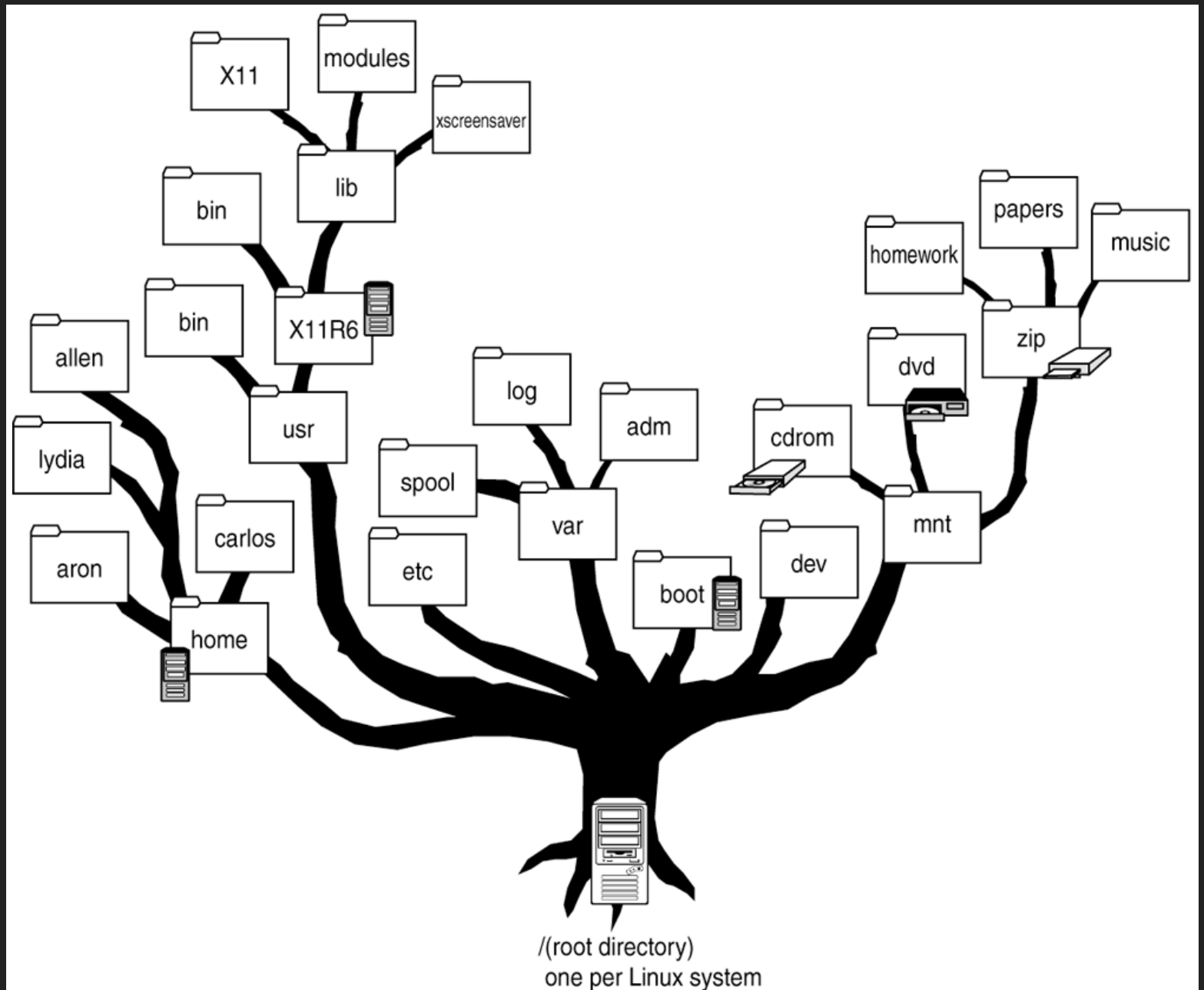
FILE SYSTEM BASICS

DATA REPRESENTATIONS AND ACCESSES

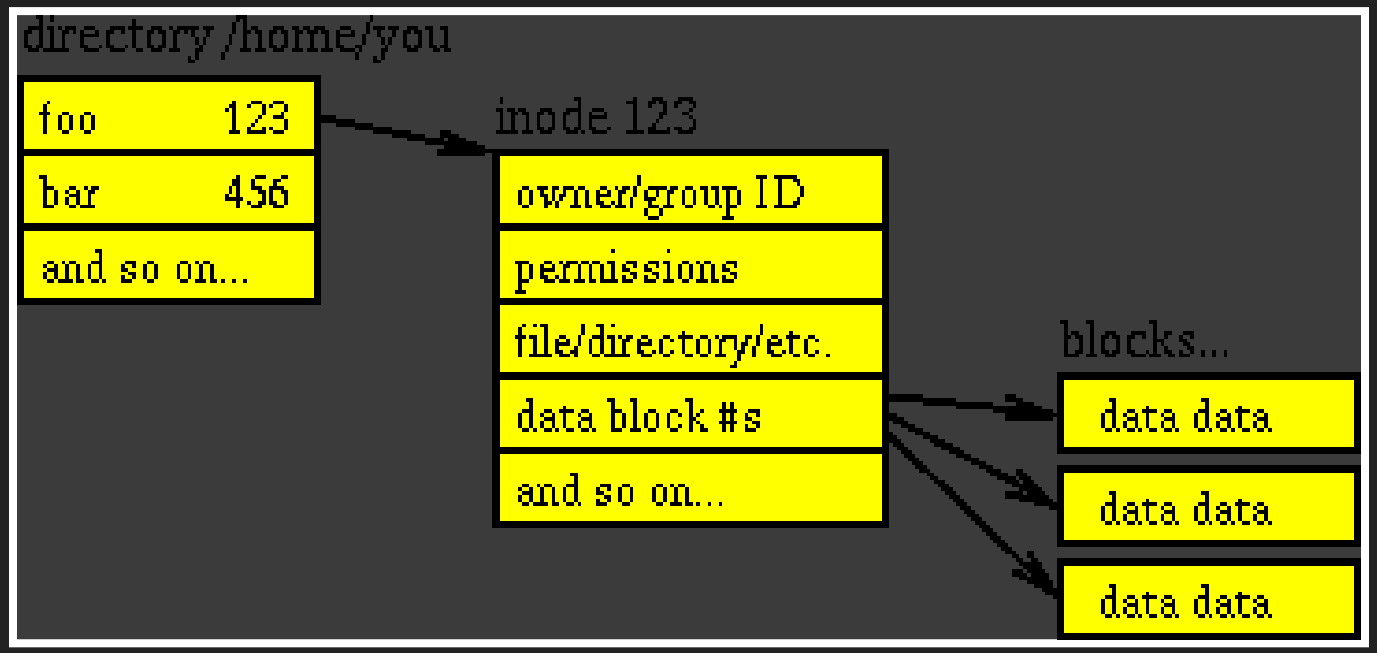
- Block storage
- File systems


User application
User application
User
Kernel
Virtual File System
Individual file systems
(ext4, vfat, etc)
Buffer/page cache
Block layer

TREE-LIKE STRUCTURE



INODE


directory /home/you
foo 123
bar 456
and so on...
inode 123
owner/group ID
permissions
file/directory/etc.
data block #s
and so on...
blocks...
data data
data data
data data

BLOCK STORAGE

Close to the storage hardware. Consider raw volumes of storage

- (-) requires specific implementations
- (+) more efficient, specific features

used for instance in databases and virtual machines

FILE SYSTEMS

Most commonly used in storage systems

Contained in block devices (*mkfs()*)

- (-) level of abstraction, worse perf
- (+) simple to use, no need to re-implement softwares

e.g. ext4, xfs, btrfs

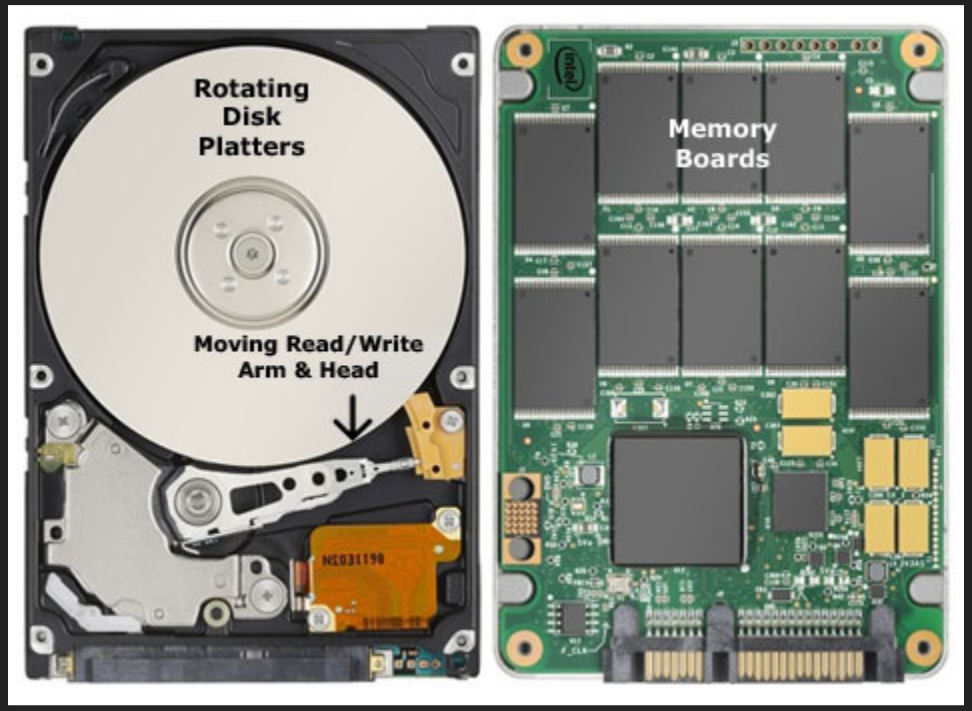
APPLICATION WORKLOADS

ACCESS MODES

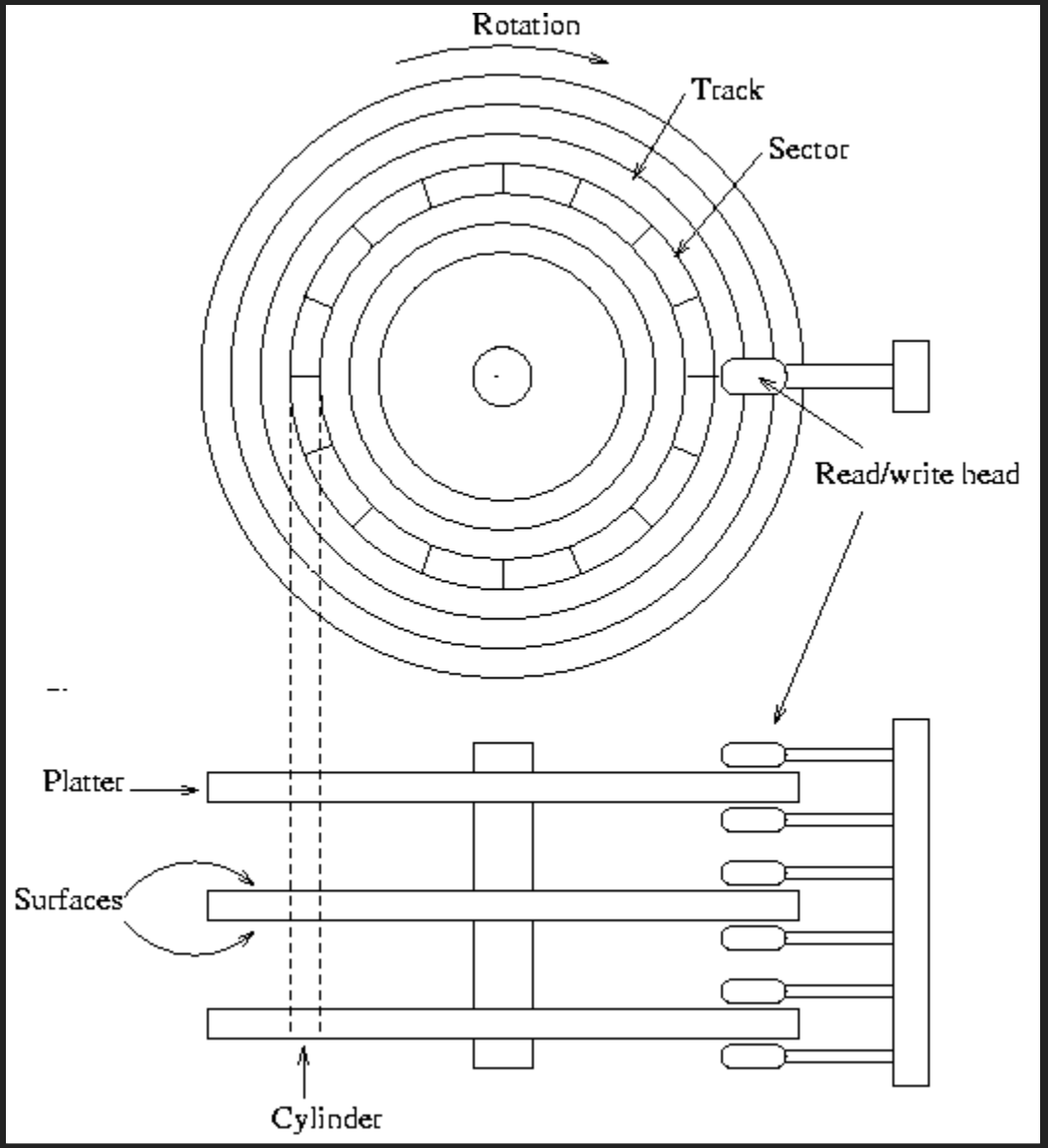
1. Type of operations
   - write
   - read
2. Access pattern
   - sequential
   - random
3. The workload depends on the application
   - video streaming: sequential over large blocks (64 KB)
   - databases: random over small blocks (4 KB)

STORAGE MEDIA

HARD DRIVE DISK (HDD)



HARD DRIVE DISK (HDD)



SOLID-STATE DRIVE (SSD)

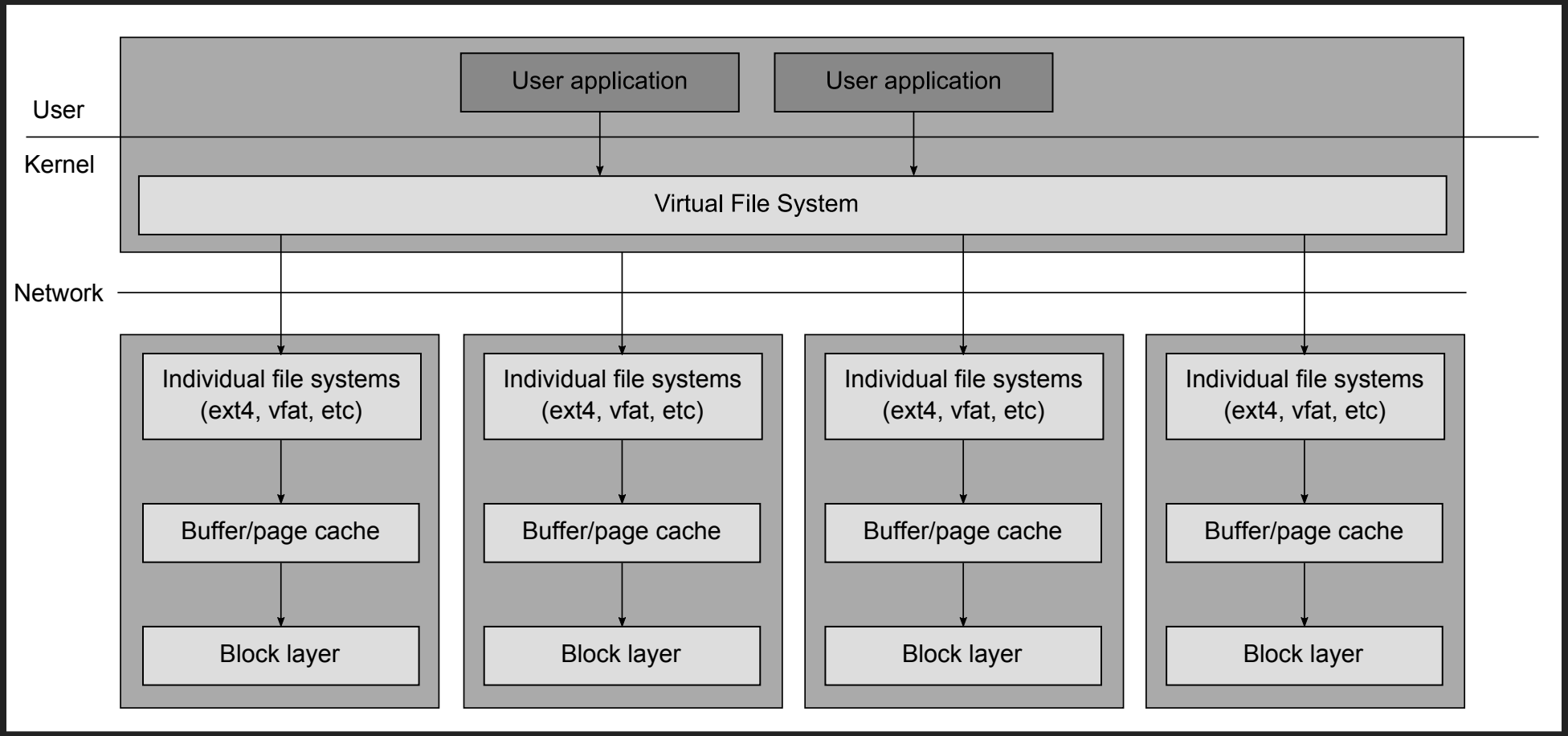
DATA SHARING OVER THE NETWORK

- Storage Area Network (SAN)
  - access block devices through network
  - *Fibre Channel* (expensive, dedicated hw)
  - iSCSI (SCSI over TCP/IP)
- Network Attached Storage (NAS)
  - file system access through network
  - NFS, CIFS, AFP

SCALE-OUT NETWORK ATTACHED STORAGE

ABSTRACTION ON TOP OF NATIVE FILE SYSTEMS

DISTRIBUTED ARCHITECTURE

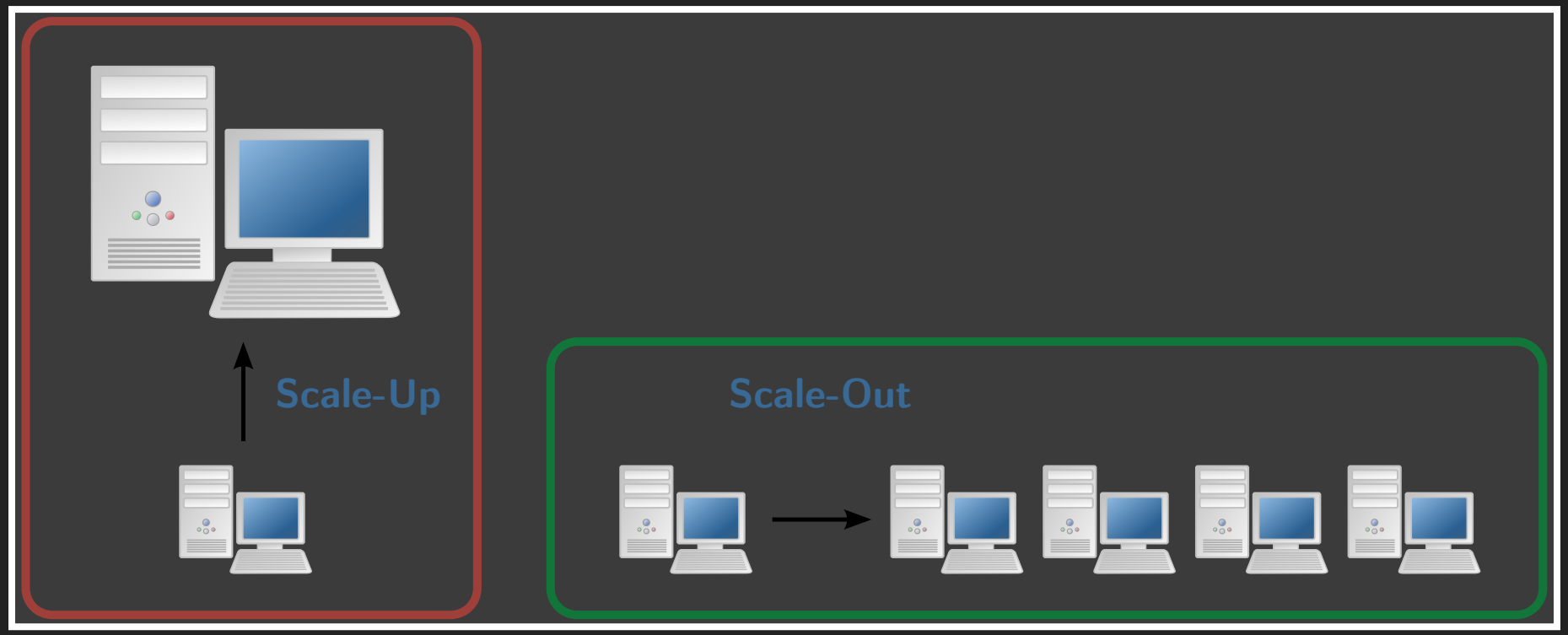


ADDRESS SPACE REMAPPING

- location independence by abstracting physical location
- virtualization system presents to user a logical space
- provides a unique namespace
- mapping for data distribution
- metadata managed within a metadata server

SCALABILITY

SCALE-UP VS SCALE-OUT



METADATA MANAGEMENT

Different implementations

- mapping table
- compute locations using an algorithm (CRUSH)

DATA MANAGEMENT

- The virtualization software redirects I/O requests
  1. receive an I/O to a location on logical disk (LD)
  2. translate location on LD into location on physical disk
  3. translate the request
- Data distribution

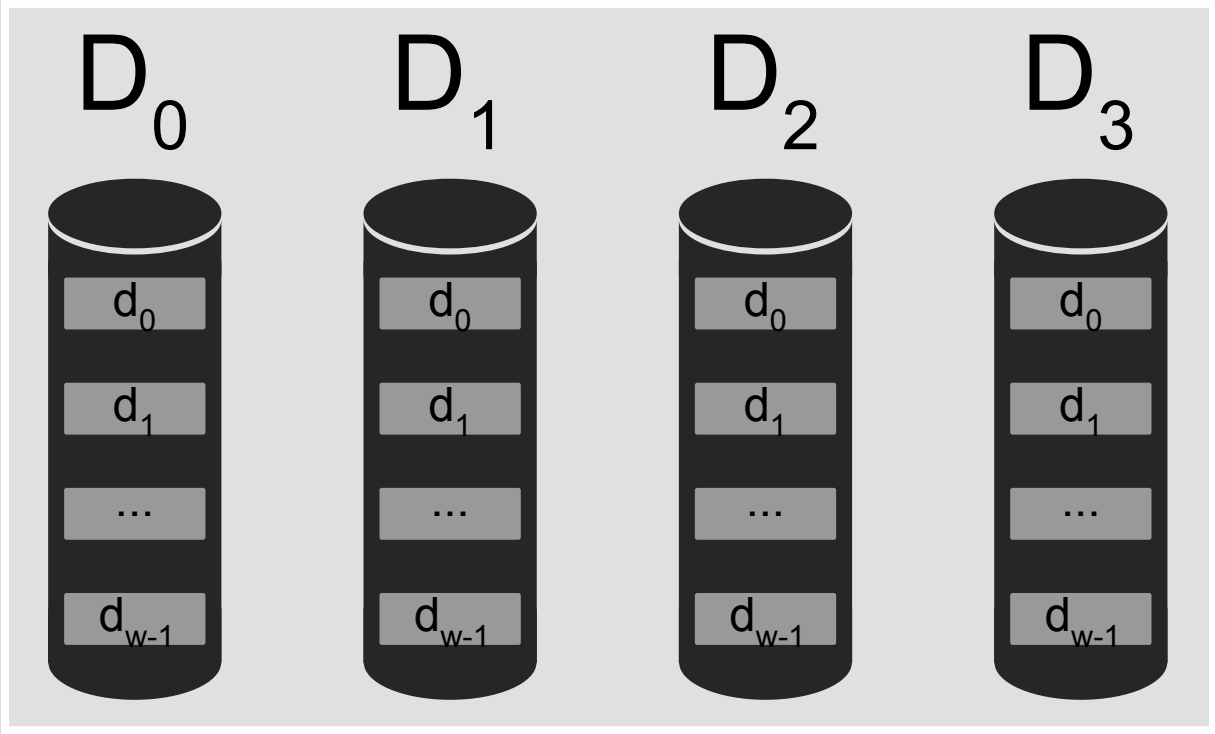
NEW DATA REPRESENTATION: OBJECT

Key/Value store

FAULT-TOLERANCE

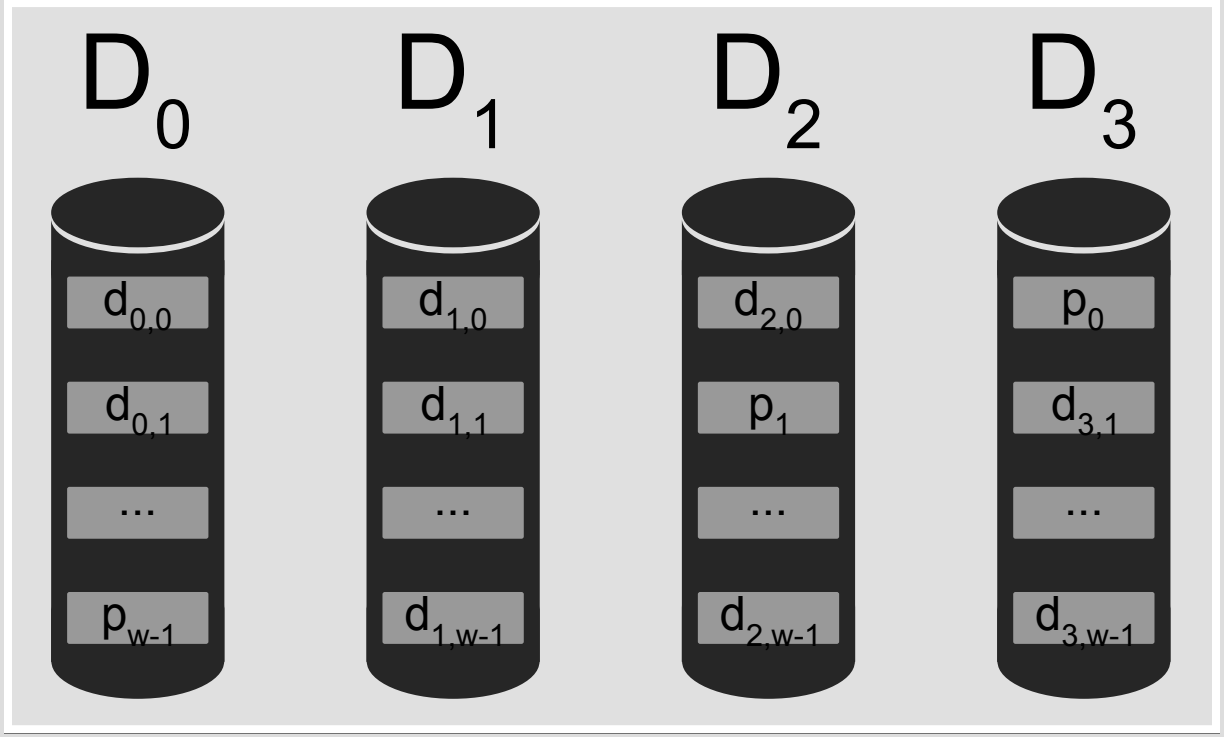

HIGH-AVAILABILITY BY REDUNDANCY

RAID-1: REPLICATION



Four-way replication here implies significant storage overhead of 300%

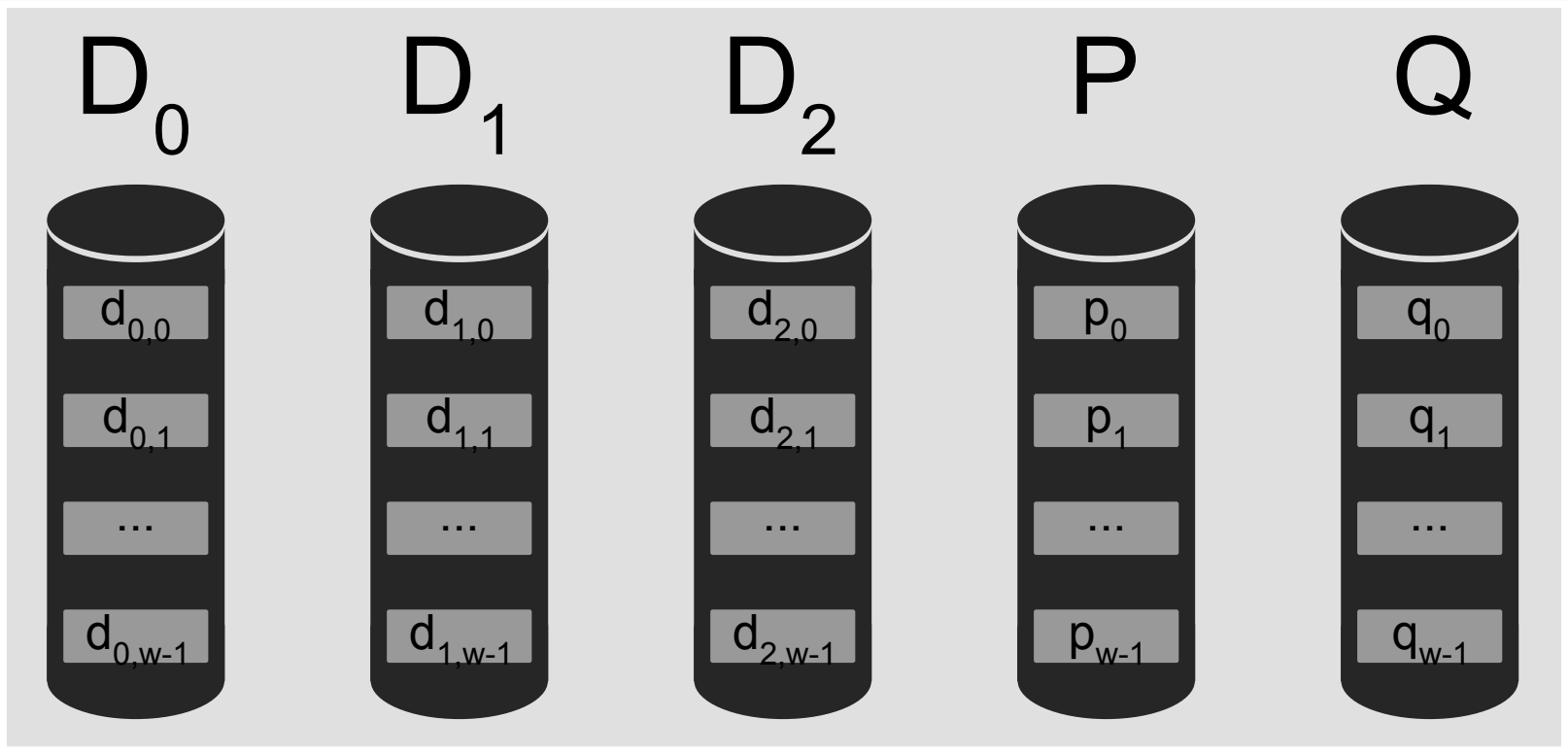
RAID-4: SINGLE PARITY



The parity disk protect the system against a single failure.

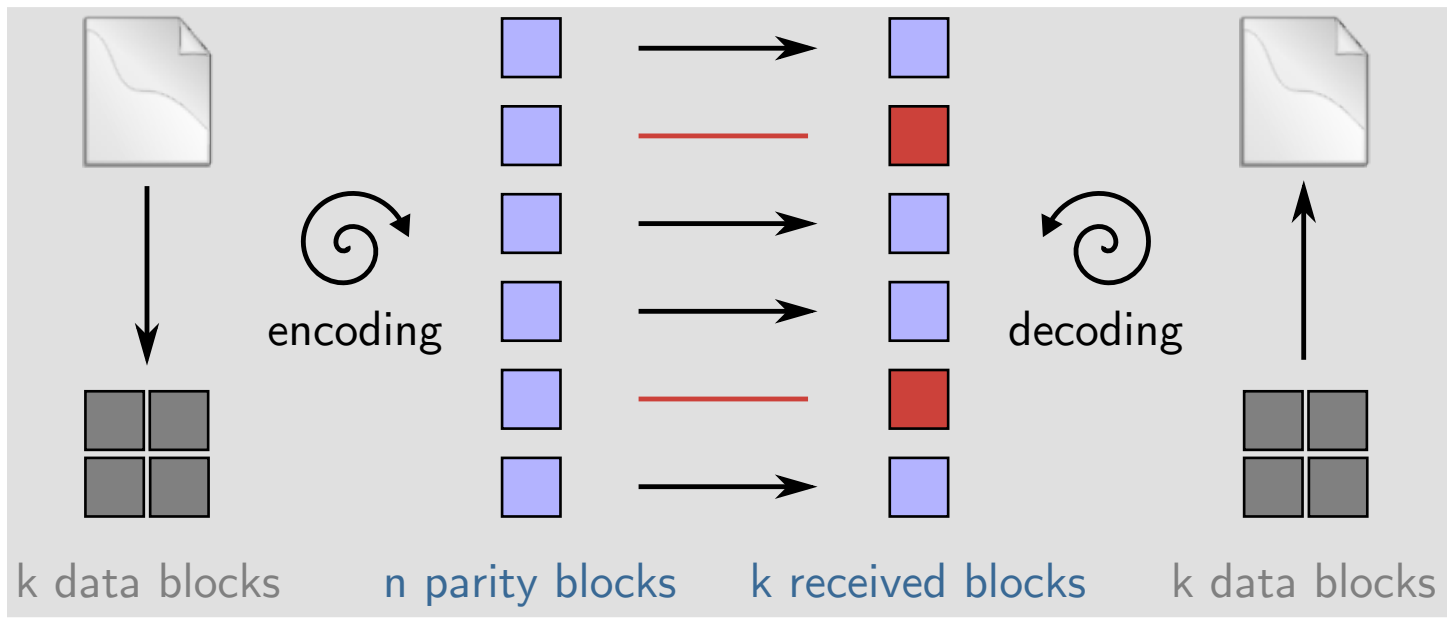
$$p_j = \oplus_{i=0}^{k-1} d_{i,j}$$

RAID-6: DOUBLE PARITY



The two parity disks protect the system against two failure.
Multiple implementations exist to compute Q.

BEYOND: (N,K) ERASURE CODE



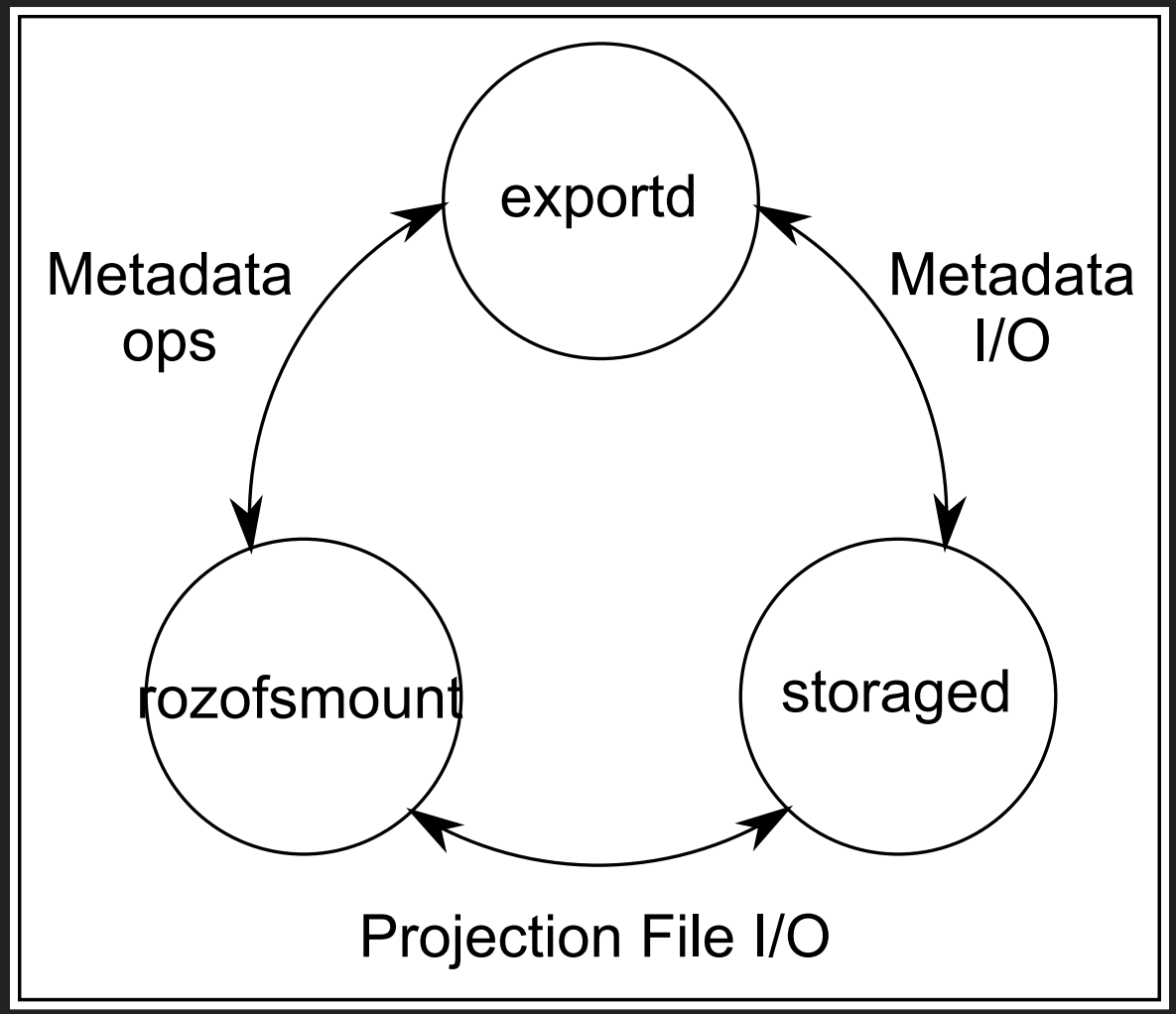
An $(n = 12, k = 8)$ erasure code can protect against up to four failures while providing only 50% of storage overhead

ROZOFS: SCALE-OUT NAS

OPEN-SOURCE DISTRIBUTED FILE SYSTEM

ROZOFS ARCHITECTURE

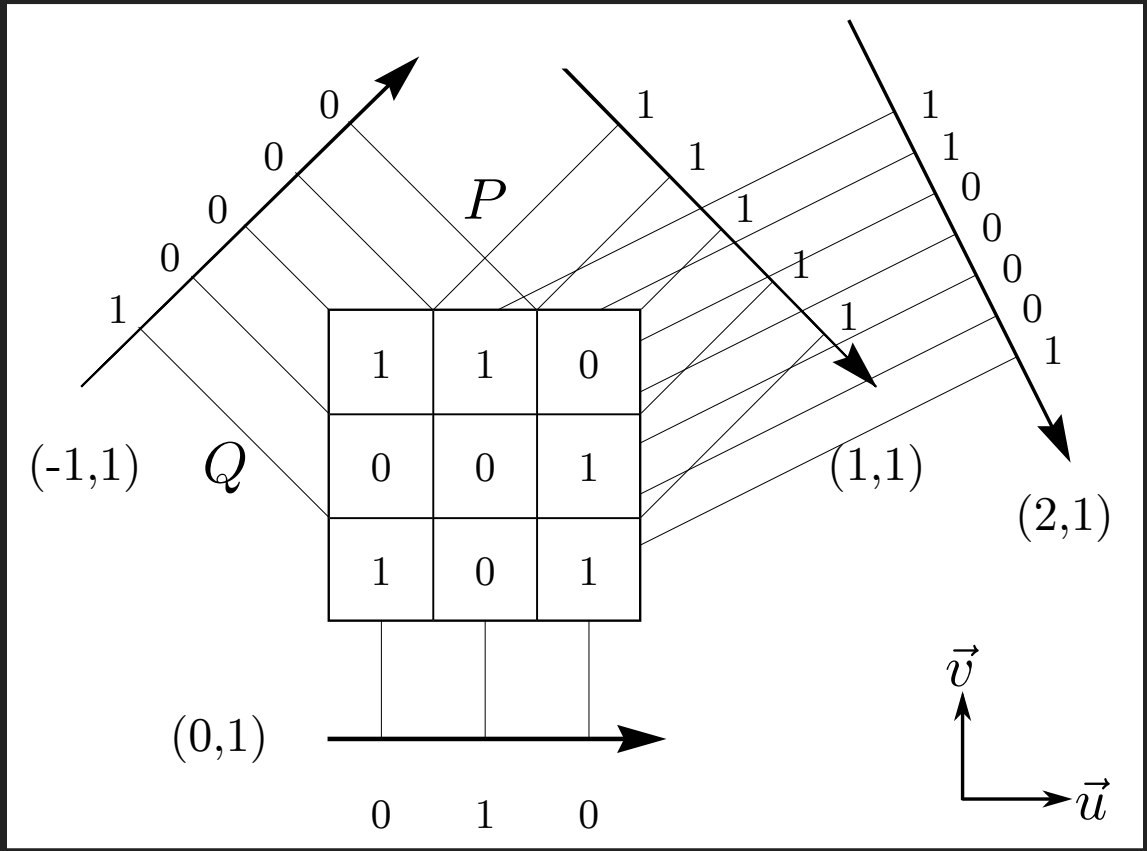
THREE COMPONENTS



FAULT-TOLERANCE

BY THE MOJETTE ERASURE CODE

MOJETTE TRANSFORM



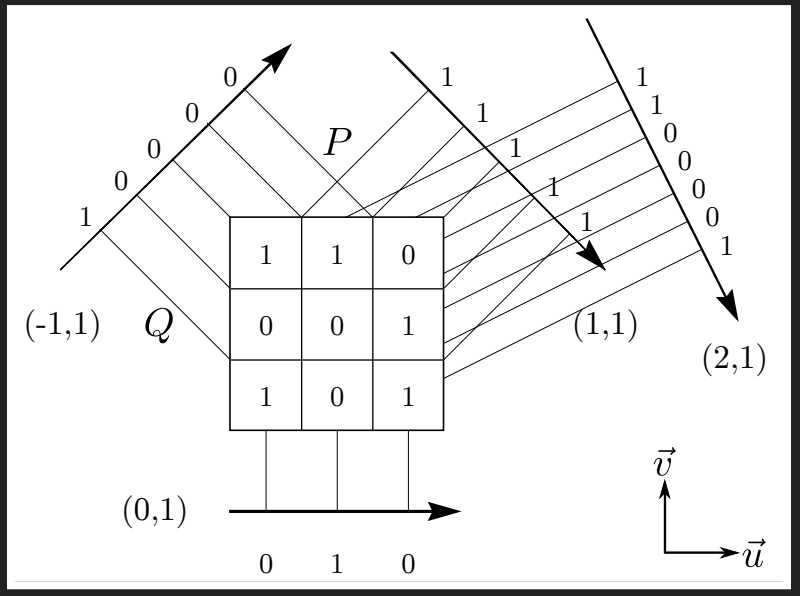
$$[\mathcal{M}f](b, p_i, q_i) = \sum_{k=0}^{Q-1} \sum_{l=0}^{P-1} f(k,l)\, \Delta(b + kq_i - lp_i)$$

KATZ CRITERION

$$\sum_{i=1}^{I} |p_i| \geq P \text{ or } \sum_{i=1}^{I} |q_i| \geq Q$$

Following simplification: $|q_i| = 1$

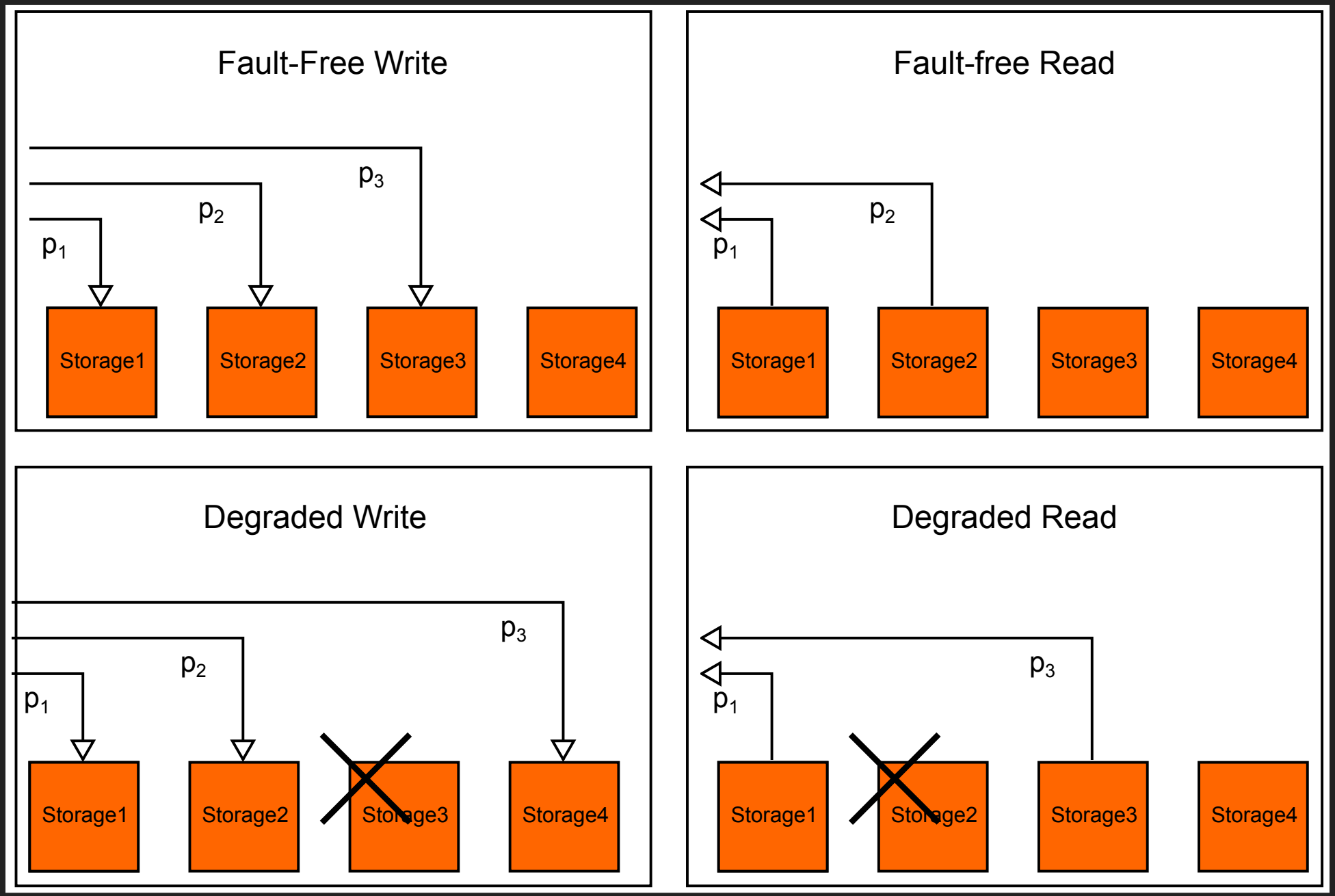
(n, k) MOJETTE ERASURE CODE



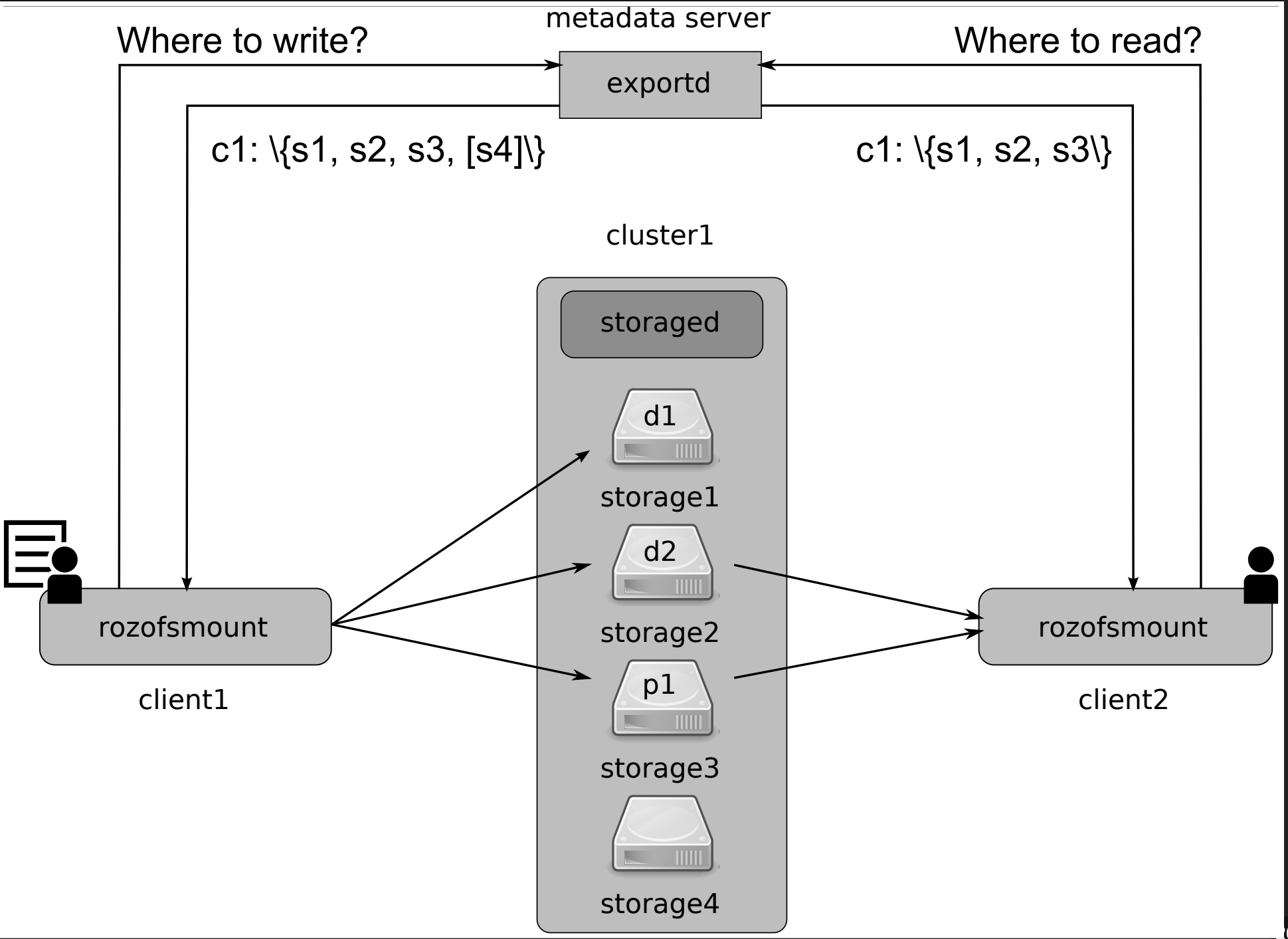
1. consider n projections, computed from $(k = Q)$-height grid
2. reconstruct from k projections
3. protection against $(n - k)$ failures

DATA I/O

READ AND WRITE SCENARIOS


Fault-Free Write
p3
p2
p1
Storage1
Storage2
Storage3
Storage4
Fault-free Read
p2
p1
Storage1
Storage2
Storage3
Storage4
Degraded Write
p3
p2
p1
Storage1
Storage2
Storage3
Storage4
Degraded Read
p3
p1
Storage1
Storage2
Storage3
Storage4

ROZOFS INTERACTIONS


metadata server
Where to write?
Where to read?
exportd
c1: \{s1, s2, s3, [s4]\}
c1: \{s1, s2, s3\}
cluster1
storaged
d1
storage1
d2
storage2
p1
storage3
storage4
rozofsmount
client1
rozofsmount
client2