Cloud Computing Applications and Services

(Aplicações e Serviços de Computação em Nuvem)

Storage

University of Minho

2024-2025

Why are storage systems relevant?

- Cornerstone for data management infrastructures and systems
  - Cloud, HPC, IoT ,...
  - Databases, Analytics, Machine Learning, Deep Learning, …
- Crucial to ensure data persistency and availability
- Performance is key
  - Slow data storage and retrieval translates into slow applications and services!

Storage Workloads

Archival

- Data is stored for archival purposes. Useful for digital information that is rarely accessed but may be relevant in the future
  - **Throughput** is favored over latency
    - Large amounts of data must be written/read efficiently
    - **Write-once** data (typically)
    - Archival files are usually append-only (i.e., no in-place updates)
  - **Sequential** workloads
    - Archival files are written and read sequentially
- Example of cloud service for archival workloads: Amazon Glacier

Storage Workloads

Backup

- Data backups of *fresh* data. Useful for digital information that is still in use and may be accessed frequently in the near future
  - **Throughput** is still favored over latency
    - Large amounts of data must be written/read efficiently
  - **Sequential** workloads (mainly…)
    - Sometimes one may want only to retrieve/update specific parts of backup files
  - **Data can now be updated** (typically in a sporadic fashion)
    - In some cases, **only diffs** (modified data) **are stored** across backups of the same data

- Example of cloud service for backup workloads: Amazon S3

Storage Workloads

Primary Storage* (not only RAM!)

- Storage support for databases, data analytics, AI frameworks, VMs ...
  - **High-throughput** and/or **low-latency** are now desirable
    - Large amounts of data may be written/read (throughput)
    - Small sized writes/reads must be done efficiently (latency)
  - **Sequential** and **random** workloads
    - The content of files may be partially accessed and out of order
  - **Data** and **metadata intensive** workloads
    - Frequent access to the content of files (data) but also to different files (metadata)
  - Data is expected to be **updated frequently**

- Example of cloud service for primary data: Amazon EBS (cloud service)

*Definition taken from: Paulo, J and Pereira, J. 2014. ***A Survey and Classification of Storage Deduplication Systems***. ACM Computing Surveys

Storage Mediums

And the main workloads these target…

- **Tape**
  - Used for archival data
  - Reliable and cheap
  - No support for random accesses or in-place updates
- **HDD**
  - Used for archival, backup, and (still in some cases) primary data
  - Still cheap, with support for random accesses and in-place updates
- **SSD** (includes SATA/NVMe SSDs)
  - Used for backup and primary data
  - More expensive than HDDs but faster, specially for random accesses
- **Persistent Memory***
  - Used for primary storage (mainly used as cache)
  - Speed closer to RAM for sequential and random workloads, but expensive
- **RAM** (Volatile!)
  - Used for primary storage (used as cache)
  - Volatile, when the computer reboots data is lost…

* Intel Persistent Memory is now discontinued

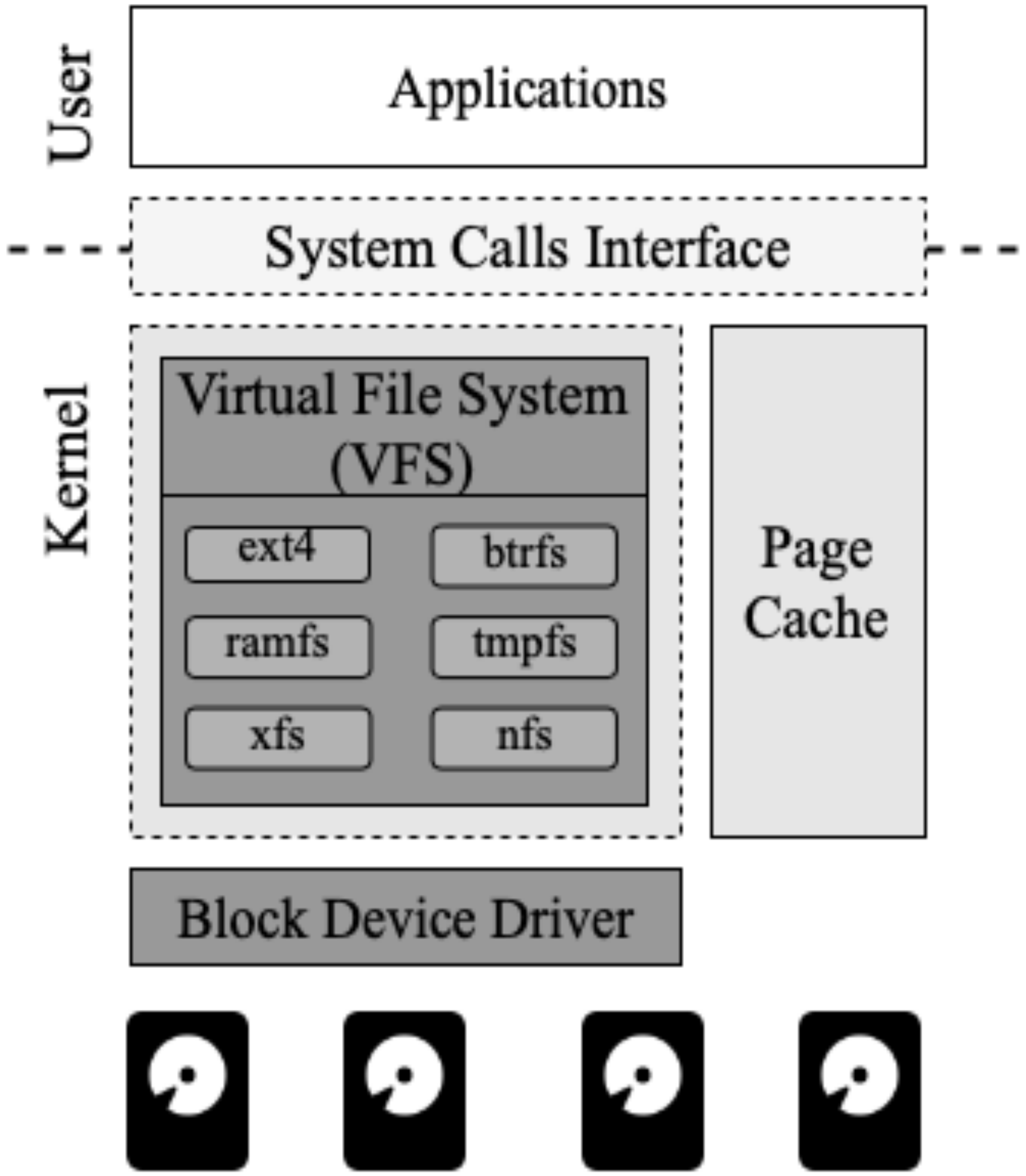
Storage Interfaces

- **Block Device**
  - Data is managed a set of blocks (closest abstraction to the disk)
  - *e.g.*, Linux block device, iSCSI, Amazon EBS, Ceph
- **File System**
  - Data is managed as a hierarchy of files and directories (abstraction that most users rely on their personal computers)
  - *e.g.*, Ext4, NFS, HDFS, Lustre, Ceph
- **Object Storage**
  - Data is managed as objects (e.g., each file is mapped to a key-value pair, the key is an unique file identifier, and the value is the file's content)
  - *e.g.*, Amazon S3, Openstack Swift, Ceph

Storage Scope

From local…

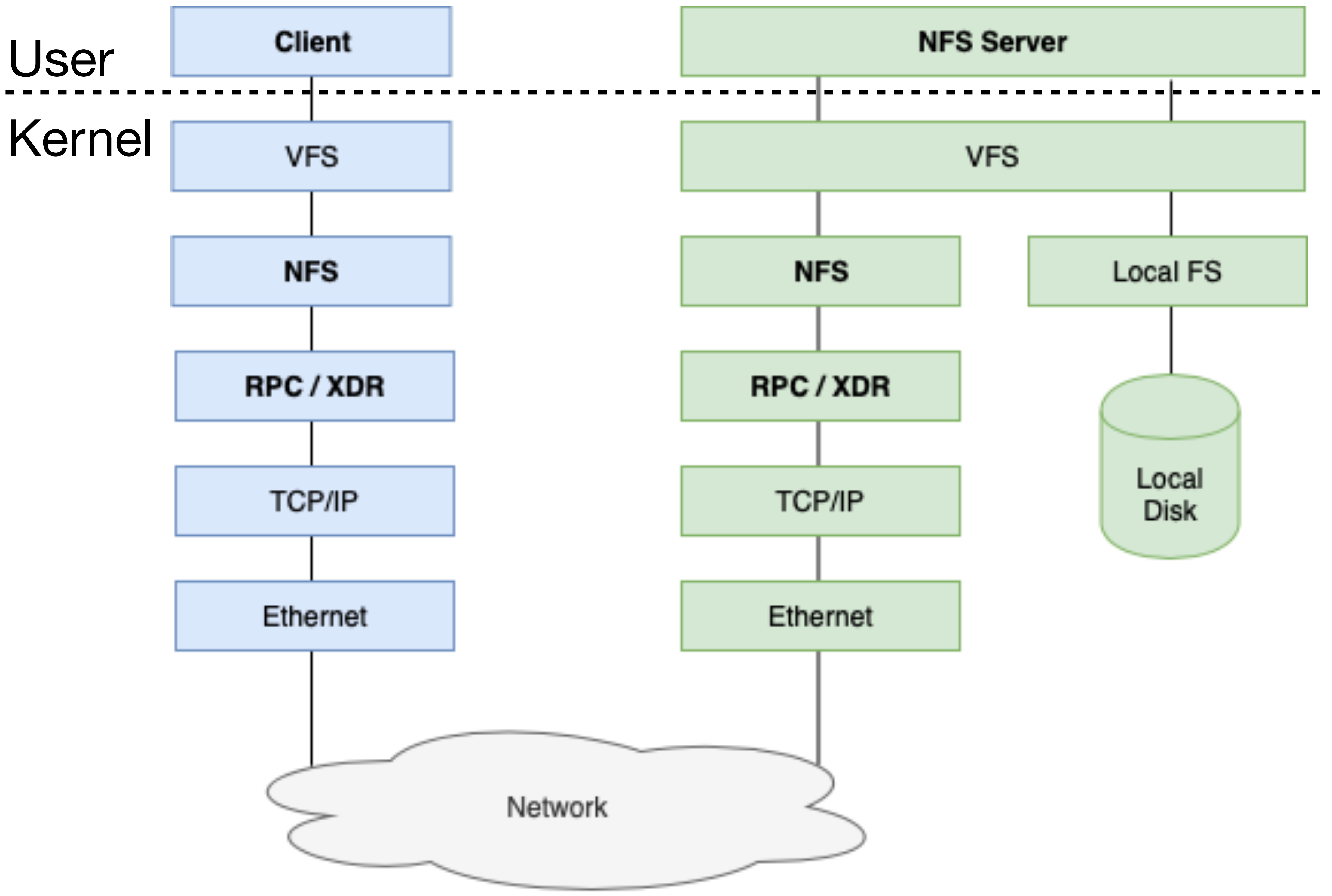
- The **Operating System (OS) mediates I/O requests** from applications to the local disk(s)
  - Remember the system calls?
- Applications can interact directly with the **block device layer** or …
- With the **file system** (e.g., ext4, ZFS, xfs)
  - The **Virtual File System** provides a common interface and abstraction
  - Different file system implementations must follow the VFS abstraction, while specializing it for their needs
- The OS **page cache** holds content of files in memory for quicker access



Storage Scope

To remote…

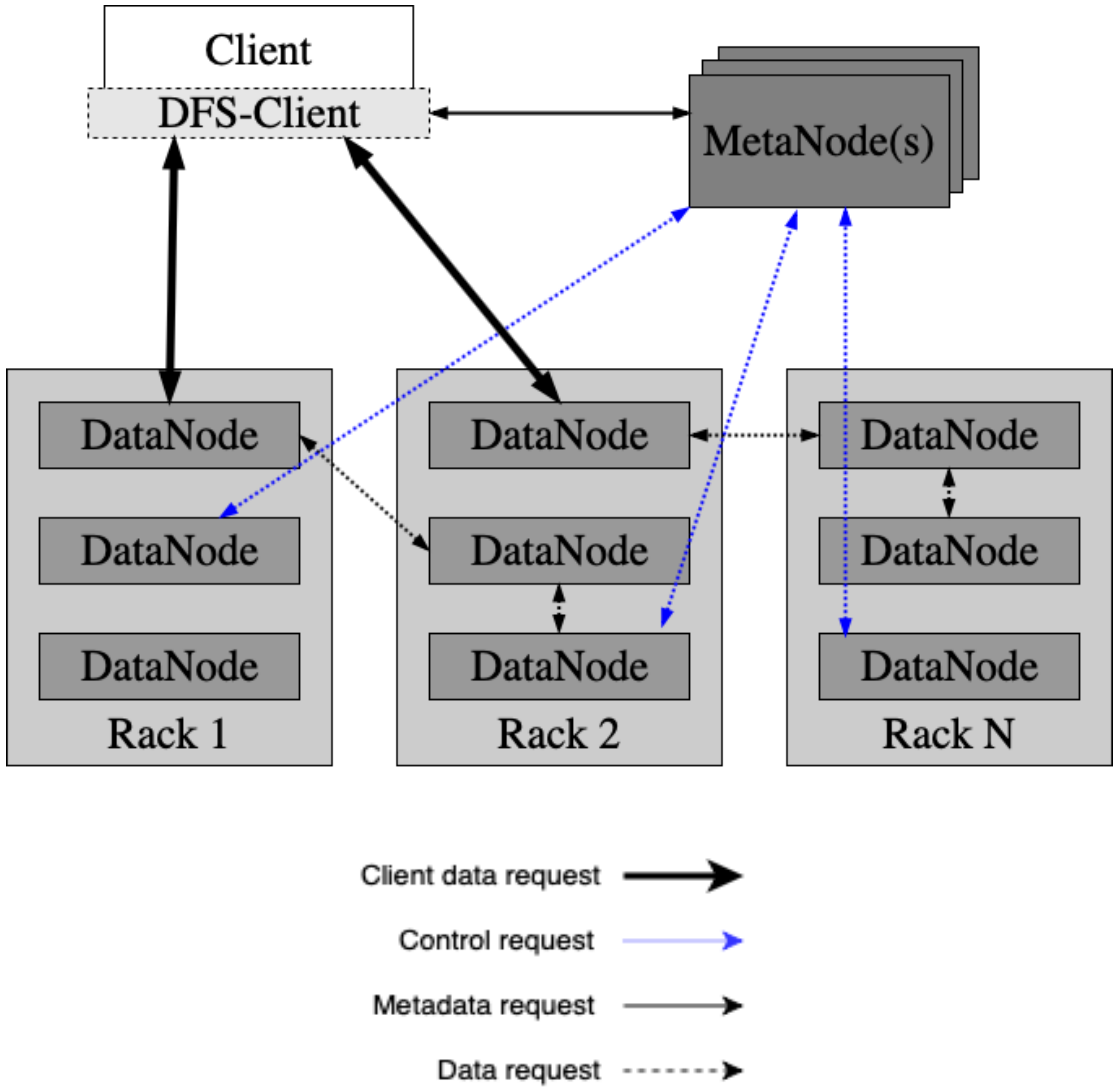
- **Storage** is provided **across the network**
  - Block Devices (e.g., iSCSI)
  - File Systems (e.g., NFS)
- **Client-server** architecture
  - Client I/O requests are intercepted (e.g., VFS and NFS specialization) and sent over the network
  - At the remote machine, requests are forwarded to the server component (e.g., NFS server) and then stored at the local disk



Storage Scope

To distributed… (data center)

- **Large-scale** (e.g., Cloud and HPC centers)
  - Tens to hundreds of nodes storing data
  - Examples: HDFS, Ceph, Lustre, GPFS
- **No single point of failure**
  - Data distributed (replicated/sharded) across “data” nodes
  - Metadata (e.g., location of files, permissions) managed by independent “meta” nodes
  - Clients contact “meta" nodes to get the location of their data. The retrieval/update of such data is done directly through the “data” nodes
- **Manager-worker design** optimized for **stable churn*** (i.e., failure of servers at a small and stable rate)

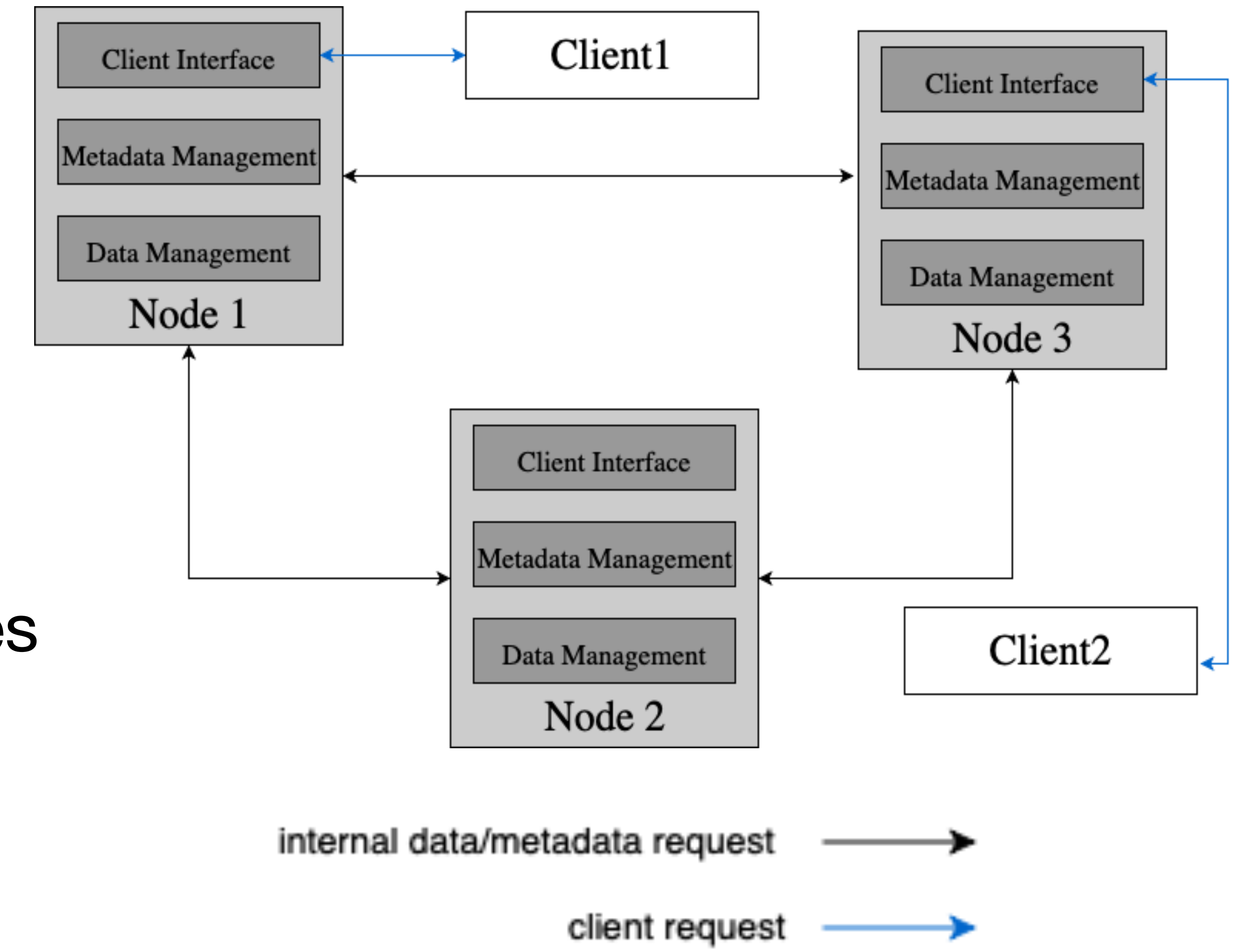


* Nodes entering and leaving the system

Storage Scope

To highly distributed... (peer-to-peer)

- **Very large-scale** (*e.g.*, IoT infrastructures)
  - Thousands to millions of nodes
  - Examples: Napster, Gnutella, CFS, DataFlasks*
- **No single point of failure**
  - Data and metadata distributed (replicated) across several nodes (i.e., no specialized nodes)
  - Clients can interact (i.e., make requests) to any node
  - Difficult to maintain a **consistent data view** across all nodes (i.e., ensuring that clients connecting to different nodes at the same time observe the same files and content)
- **Peer-to-peer design** optimized for **high churn** (i.e., nodes are expected to fail and rejoin the system frequently)

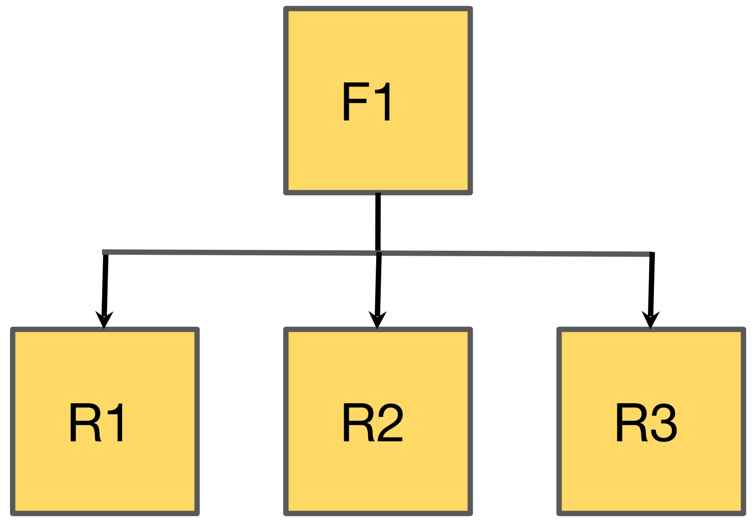


* Maia et al. 2014. *DATAFLASKS: Epidemic Store for Massive Scale Systems*. Symposium on Reliable Distributed Systems (SRDS)

Storage Features (some examples…)

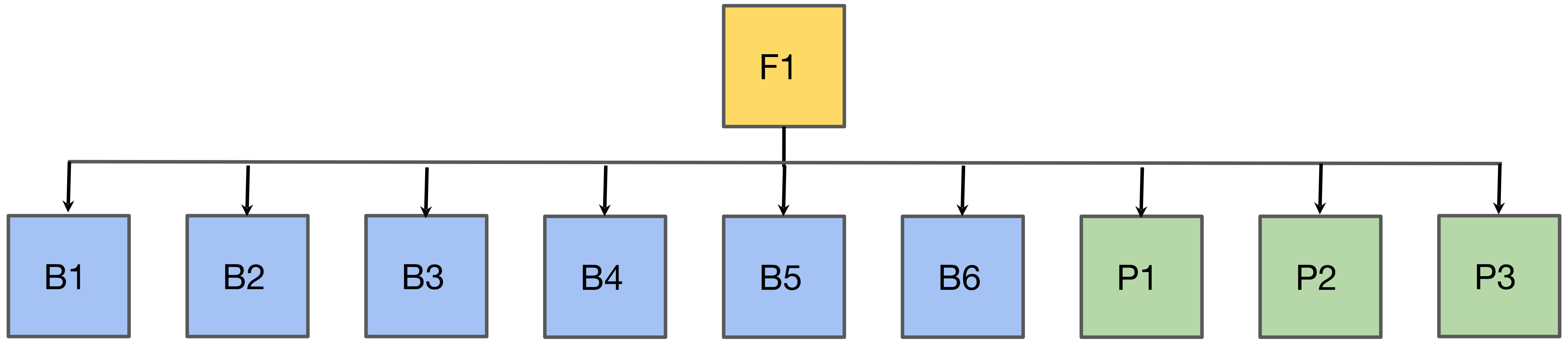
Data availability

- **RAID - Redundant Array of Inexpensive Drives:** data is replicated/sharded across multiple local disks in a single server for availability and load balancing purposes
- **Replication:** data is replicated across several servers for availability and load balancing purposes
- **Erasure-Codes:** data is broken into fragments, with redundant information, that are then spread across several servers for availability and load balancing purposes



Replication: Exact Replicas (of F1)

E.g., Replication factor = 3
- Tolerates 2 failures
- 3X storage overhead



Erasure-codes: Original data (F1) is divided into k Blocks and m Parity blocks

E.g., Reed-Solomon (k = 6, m = 3)
- Tolerates 3 failures
- 1.5 X storage overhead

Storage Features (some examples...)

Performance optimizations

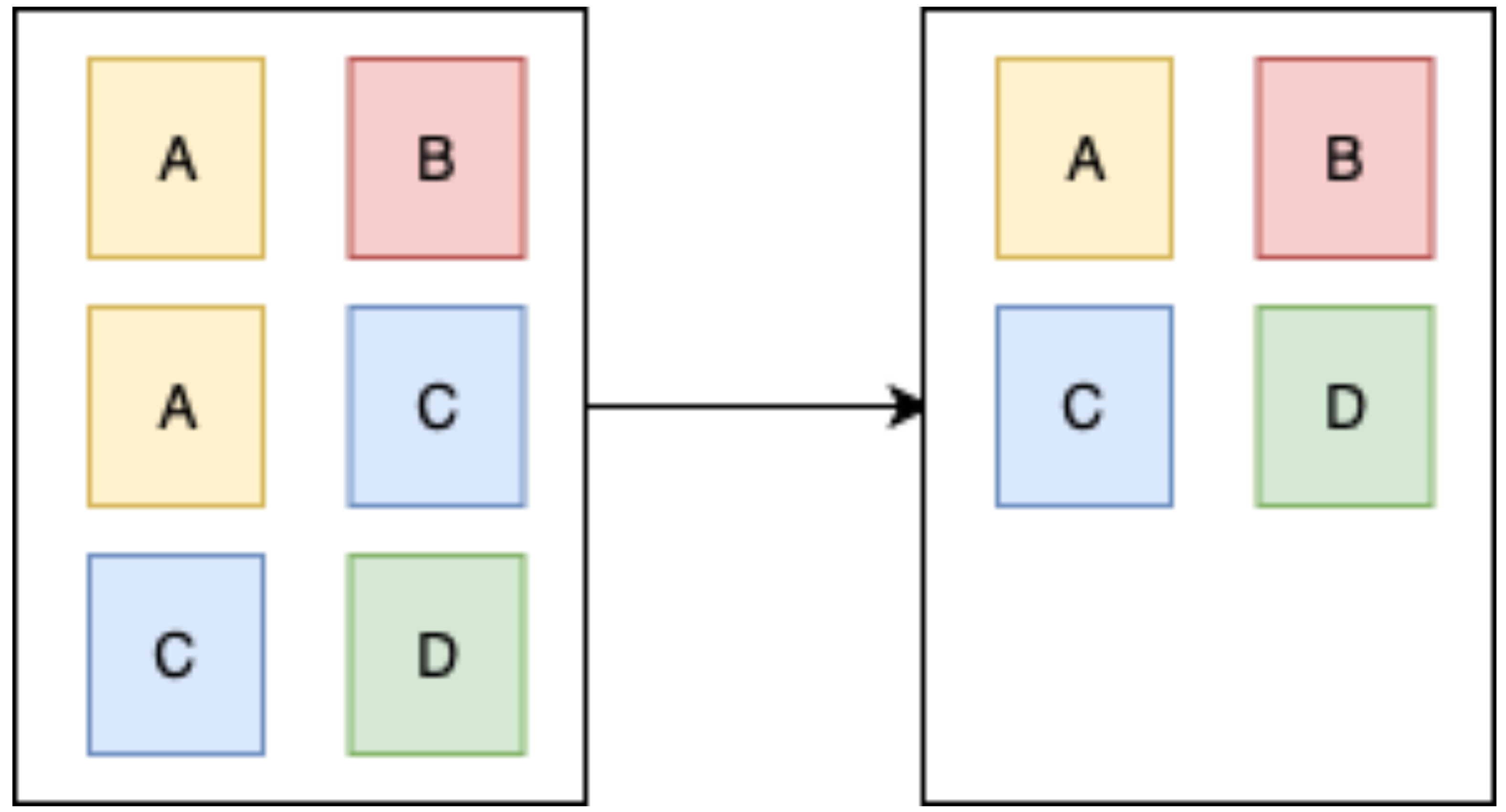
- **Data locality:** Push computation near to the devices and/or servers holding data
  - Storage and processing co-location at the same server / device, which means faster access to data!
  - Example: HBase and HDFS, active storage
- **Caching:** Keep data closer to the client and/or accessible from a faster source (e.g., RAM)
  - Avoids waiting for data to be written/read from local or remote storage
  - Example: File system page cache, Alluxio (in-memory distributed file system)

Storage Features (some examples…)

Space efficiency

- **Compression:** Removes redundant content (e.g., bytes) inside and across files
  - Usually used as a static technique
    (i.e., for sets of files that are not going to be further updated)

- **Deduplication:** Eliminates redundant copies at a storage system (e.g., files / blocks)
  - Used as a dynamic technique (i.e., some files /blocks are expected to be updated in the future)



Storage Features (some examples…)

Security

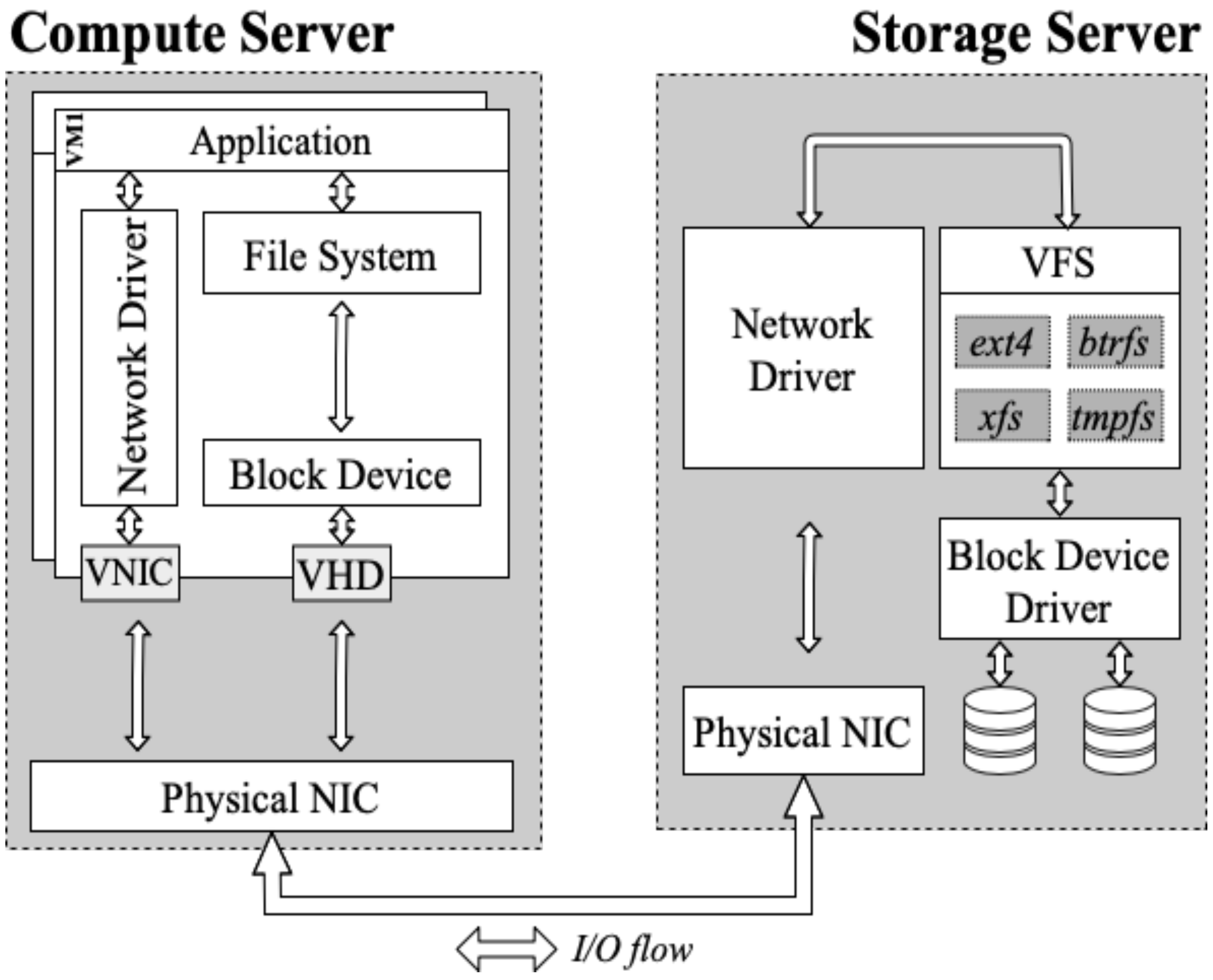
- **Data Encryption:** Privacy protection for sensitive data
  - **Encryption at rest:** Data is encrypted before being stored persistently
  - **Encryption in transit:** Data is encrypted at the client premises before being sent through the network (e.g., for remote storage systems)
- **Access Control:** Avoid unauthorized access to users data

Complex and Monolithic Storage Solutions

Modern infrastructures

- The **I/O stack** of data centers is **long** and composed of **many components**
  - applications, local file systems, VMs, remote storage, disks, …
  - Each providing a **strict combination of storage features**, however…

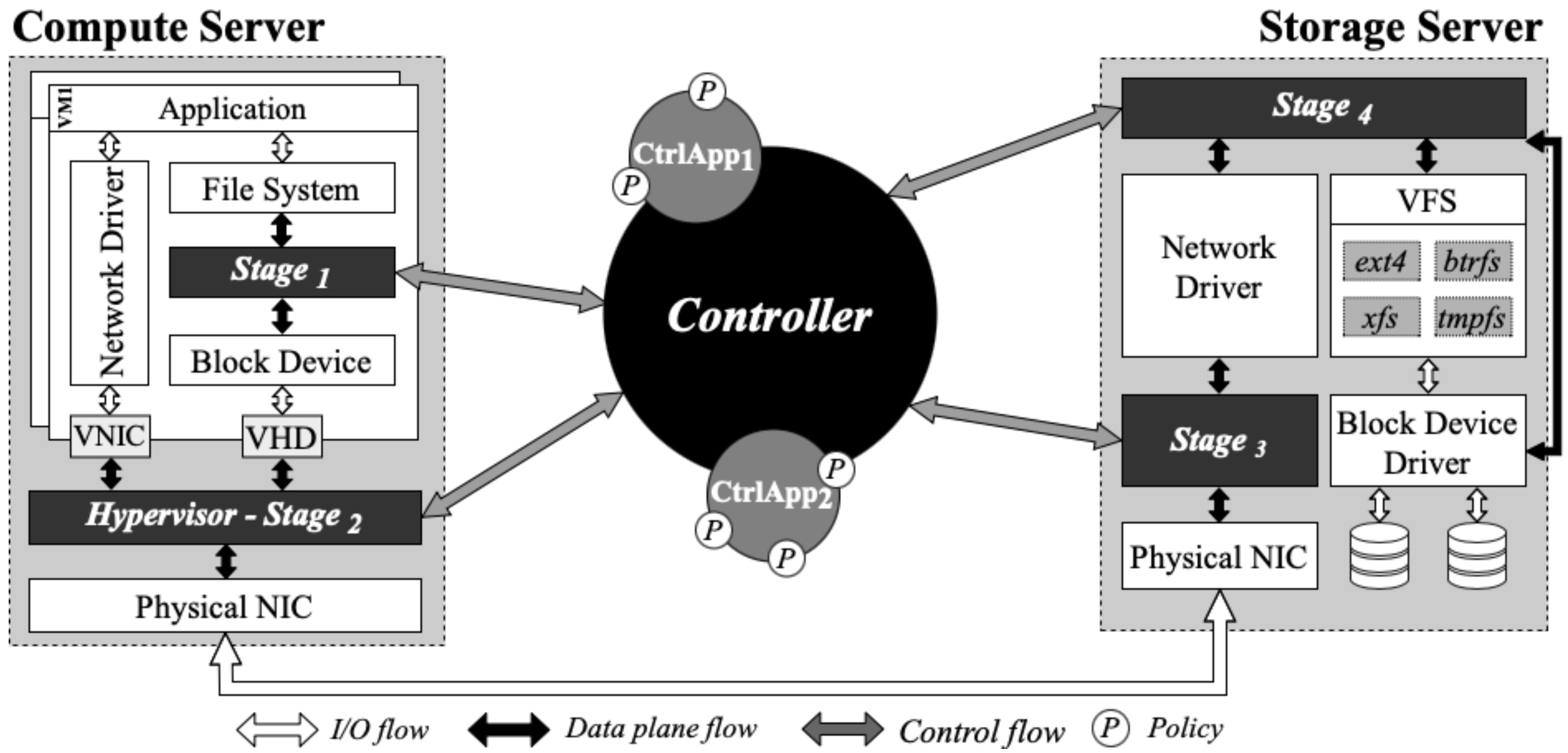
- The best combination of features to apply varies with the applications
  - Small files versus large files
  - Storage access patterns
  - Sensitive vs non-sensitive information
  - …



Software-Defined Storage

SDS

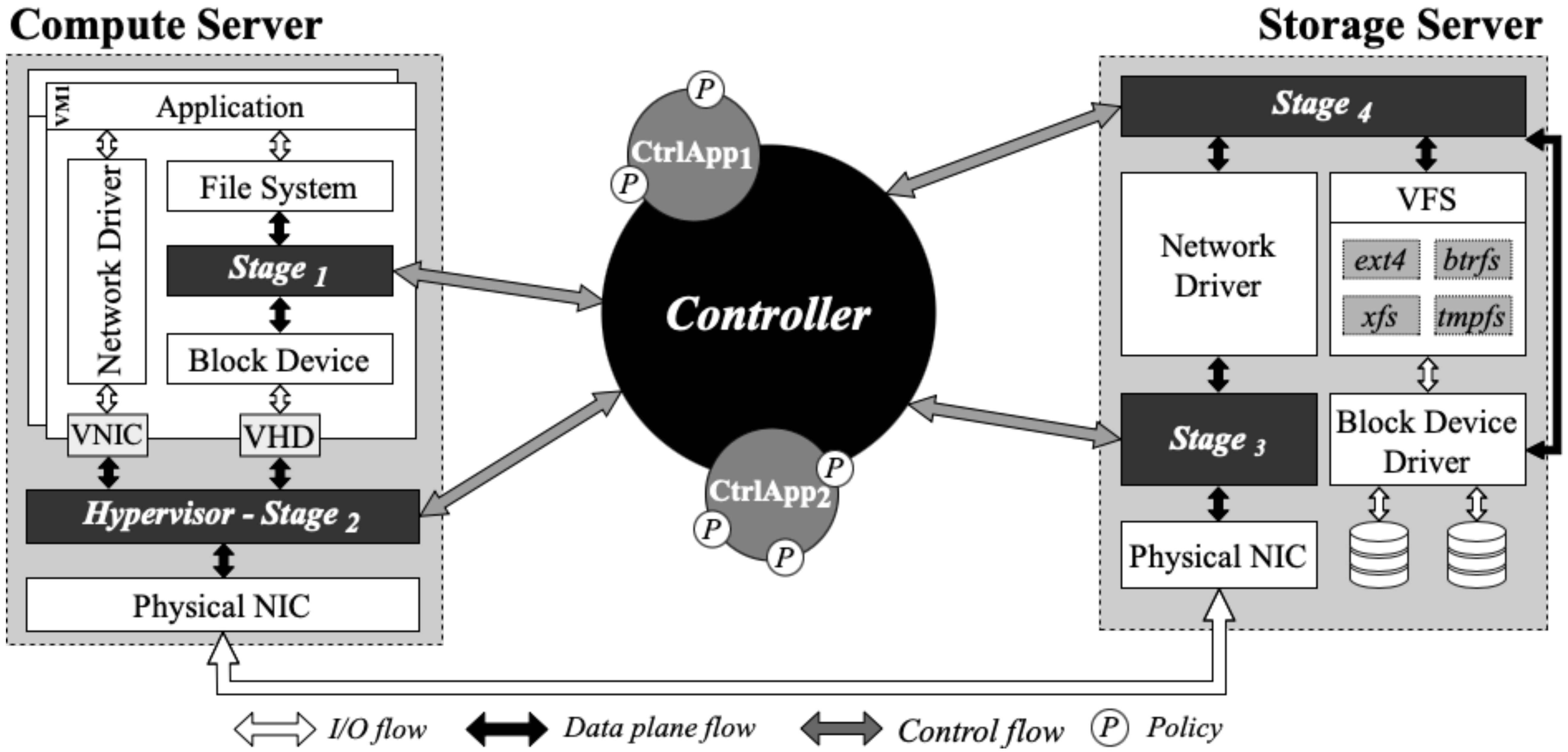
- Main principles
  - I/O flows (**data plane**) are separated from the control flow (**control plane**)
  - The control plane ensures **global control of I/O flows** (logically centralized)



Software-Defined Storage

Data Plane

- **Layered** design organized into **stages**
- Each **stage** handles requests at the I/O path and provides different features
  - Examples: caching, compression, encryption
- **Programmable** and **extensible** design, i.e., stages can be extended with new features



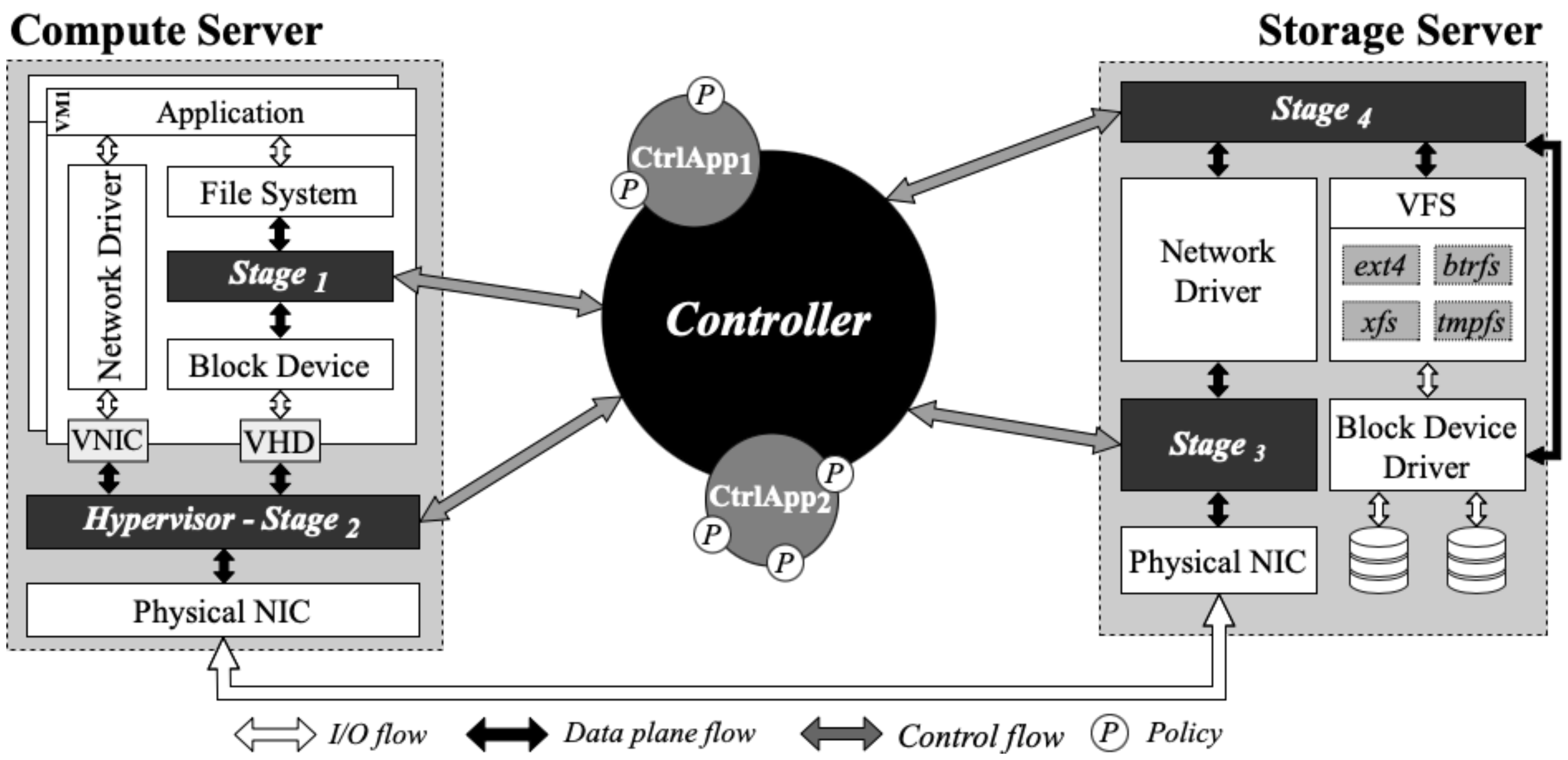
Software-Defined Storage

Control Plane

- **Global visibility** of applications, stages and infrastructure resources
  - The brain of the system that holistically coordinates data plane stages

- **Distributed** for scalability and availability purposes

- Configures and tunes data plane stages to enforce I/O policies
  - **Quality of Service** (*e.g.,* I/O fairness or prioritization)
  - **Transformations** (*e.g.,* encryption, compression)
  - Policies are defined through **Control Applications**



Further Reading

- Macedo R, Paulo J, Pereira J, Bessani, A. 2020. ***A Survey and Classification of Software-Defined Storage Systems***. ACM Computing Surveys
- Paulo J, Pereira J. 2014. ***A Survey and Classification of Storage Deduplication Systems***. ACM Computing Surveys
- Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. 2006. ***Ceph: a scalable, high-performance distributed file system***. Operating Systems Design and Implementation (OSDI)
- Maia F, Matos M, Vilaça R, Pereira JO, Oliveira R, Rivière E. 2014. ***DATAFLASKS: Epidemic Store for Massive Scale Systems***. Symposium on Reliable Distributed Systems (SRDS)

Questions?