Storage Systems

NPTEL Course
Jan 2012
(Lecture 01)

K. Gopinath
Indian Institute of Science

Introduction

- Why Storage Systems?
  - Earlier: (processing + storage) and networking
  - Now: processing, storage and networking
  - Fast networks enable separation from “processing”
- Devices, Protocols, Layers/Systems
  - Old and New Devices: Tape, Drum, Disk, Solid State
  - Protocols: NFS, Cloud storage API
  - Layers/Systems: Google FS, Mail storage
- Issues
  - Older: concurrency with CPU, handling device diversity
  - Newer: scale, distribution, error mgmt, security, RT, QoS, manageability

Why is storage different?

- Consider long term storage (multiple decades): Stored data can be accessed decades later!
  - Formats, devices, etc can change
  - Data not interpretable unless auxiliary information also stored (Recursion problem!)
- Consider security: Why storage security different from, say, network security?
  - Network security is across “space”
    - Network transfers happen within a short time (unless space probe netw packets to Pluto!)
  - Storage security is across both “space and time”
    - If using keys, keys may have to survive years!

Naming and Storing

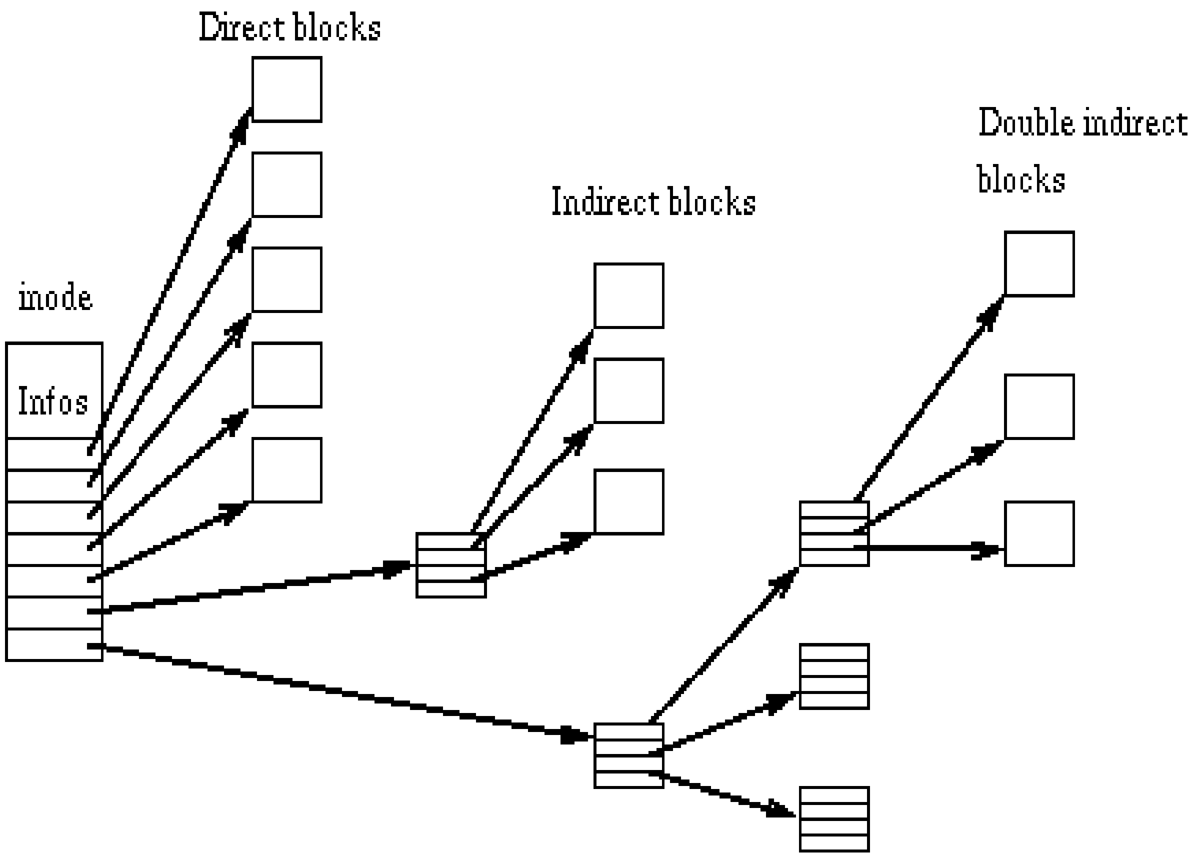
- Two important functions:
  - Give a name to an object
    - Can involve some processing (eg. add name to an index)
    - Or, name itself may be computed based on contents
  - Store an object
    - May involve other reads or stores!
    - May involve significant computation
      - Compression, encryption, coding, or deduplication: remove redundancy
- May need to keep aux. information about object
    - Metadata vs data; recursion? (metadata about metadata...)
    - Metadata loss vs data loss
- Access (r/w): device specific aspects determine speed
  - Reads sequential, non-sequential or random
  - Writes in-place or out-of-place

Large persistent data structures

- For processing, parts need to be brought into mem.
  - Two copies: “in memory” and “on-disk”
  - Atomicity and Consistency issues
- Algorithmic aspects need to be carefully taken care of
  - Number of objects can be in billions
  - Size of object can be in gigabytes
  - As time progresses, newer algs needed as scale changes
    - Mail directories can have 1000's of msgs, each a file!
    - Creating a file requires locking directory
    - Concurrent creates to same directory may become lock-bound
  - Critical with web-level storage systems
  - Many newer models (eg. key-value stores) since c. 2000

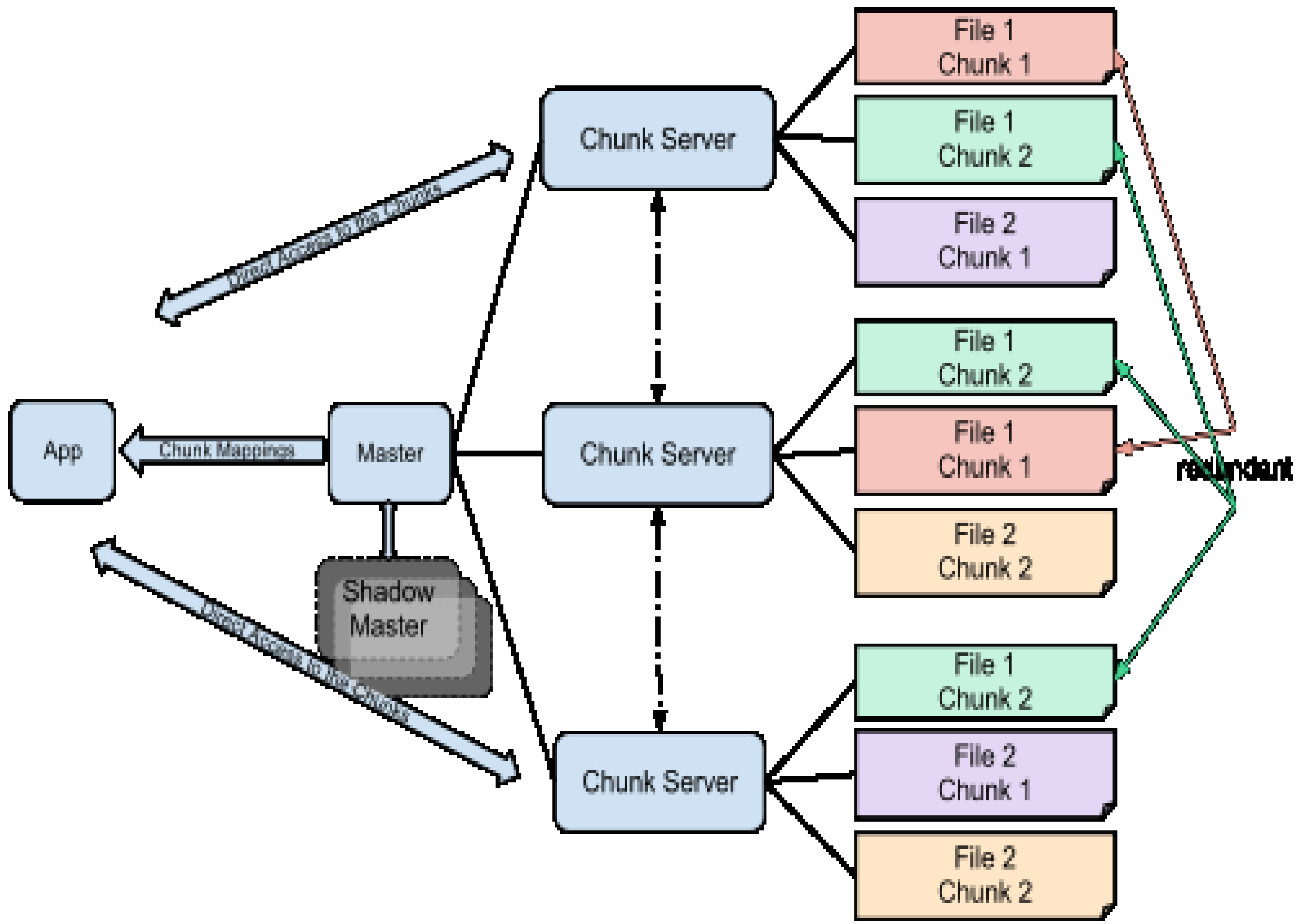
Ext2 FS

(from Wikipedia)



Google FS

(from Wikipedia)



Deep Storage Stack

- Various types of abstractions in stack
  - Device
  - Block
  - File
  - Application level (buffering in libc, for eg.)
- Finer sublayers in each layer
  - SCSI has upper (device-specific), mid (protocol specific) and lower (physical communication layer)
- For scalability, network stack part of storage stack
- A good part of stack in kernel
  - Increasingly, storage stack migrating out of kernel with separation of processing and storage (eg. GFS)

Storage Characteristics

- Concurrency arises naturally
  - Wide disparity in speeds of memory and storage
  - Have to mask slowness of storage
  - Make processing and storage go at their own rates and use interrupts (or sometimes polling) to signal completion of slow storage operations
  - Historic reason why operating systems developed
- Storage has to be typically persistent over time
  - Amount increases typically with time
  - But not all imp over time; keep imp part in fast storage?
- “Caching” and “tiering” arise naturally
  - Have to choose 2 of 3: speed, capacity or cost

Storage Performance

- Storage often slowest component
  - Cache!
- Within single device, efficiency by:
  - merging requests
  - scheduling requests in an order that is best wrt device (out-of-order execution commonplace)
    - Higher level software has to work around this aspect
    - If a particular order required, left to “user”
    - Semantically not much guaranteed
- Asynchronous processing often used: aio
- Parallelism across multiple devices/threads:
  - Multiple Heads (Disks)
  - Multiple chips (SSDs)

Optimization Framework

- Due to slowness of devices, optimization of accesses important
  - eg. what to cache, what to prefetch?
  - But usage patterns typically not known *a priori*
  - Big difference in performance whether sequential access or random
  - System slow if too many on demand migrations from slow to fast tier of storage (latency delays)
- Often, opts. critical and override “semantics”
  - Out-of-order processing typical
  - Complex higher-level software
- Learning on the job important
  - Simple and robust methods useful

Storage Protocols

- Interrupt driven rather than wait/poll
  - On completion, interrupt CPU or HBA
  - To avoid interrupt overhead, HBA or similar agents
    - Helps Segmentation and Reassembly (SAR)
- Split-phase transactions common
  - for eg: on completion of (a long) seek, slave takes bus
- Protocol endpoints preferably “virtualizable”
  - SCSI devices can be on an electrical bus, network or Internet if physical layer handled correctly
    - Protocols survive much longer
  - Devices can have arbitrary structure as long as they speak SCSI protocol
    - Even big servers!

Summary

- Storage systems design has many ramifications for the rest of the system
  - Provide abstractions based on application needs and devices
  - Design needs to be sensitive to cost, devices, manageability
  - Introduce newer abstractions with time
    - eg. key value stores
- Storage systems need to scale to support large scale computing systems