Review

CS 4513

File Systems - Partitions

- What is a hard disk partition?

File Systems - Partitions

- What is a hard disk partition?
  - *Physical (or logical) storage space division on disk*
  - *Typically, so can put a file system inside the partition*
  - *Contains separate collections of directories*

File Systems - Partitions

- What is an MBR/GPT for?

File Systems - Partitions

- What is an MBR/GPT for?
  - *Contain code for boot loader so BIOS can load*
  - *Contain partition table (with partition information)*

File Systems - Partitions

- How does an OS get access to a file system?

File Systems - Partitions

- How does an OS get access to a file system?
  - *File system is mounted by OS*
  - *Mounting reads file system information from superblock, where superblock provides details on layout of file system data (e.g., free space, file descriptors…)*

File Systems - File Descriptors

- What is a file descriptor?

File Systems - File Descriptors

- What is a file descriptor?
  - *A handle/name/pointer that provides access to the blocks of data associated with a file*

File Systems - File Descriptors

- What is *good* about storing files as contiguous blocks? What is *bad*?

File Systems - File Descriptors

- What is *good* about storing files as contiguous blocks? What is *bad*?
  - Good: *file descriptors are simple (number + length)*
  - Good: *reading whole file can be efficient*
  - Bad: *Changing file size after creation problematic. Fragmentation (internal and/or external)*

File Systems - File Descriptors

- Why is a file allocation table (FAT) better than a pure linked-list when storing disk blocks?

File Systems - File Descriptors

- Why is a file allocation table (FAT) better than a pure linked-list when storing disk blocks?
  - *FAT separates linked-list from disk blocks, allowing links to be traversed in memory rather than reading from disk*

File Systems - File Descriptors

- What is an inode?

File Systems - File Descriptors

- What is an inode?
  - *A (Unix) file descriptor containing attributes for file, and pointers to disk blocks (and indirect block pointers)*

File Systems - Directories

- How are directories similar to files? How are they different?

File Systems - Directories

- How are directories *similar* to files? How are they *different*?
  - *Similar – both contain data, accessed through obtaining file descriptor via "open", then "read"/"write" and "close"*
  - *Different – access to contents (data) restricted to specific OS systems calls (e.g., readdir()), and data format/structure is specific to file system*

File Systems - Directories

- Where are file attributes stored?

File Systems - Directories

- Where are file attributes stored?
  - *It depends. Attributes can either be stored on the disk (generally bad since slow), with the file descriptor (e.g., an inode) , or with the file name in the directory entry*

File Systems - Aliases

- What is an *alias* in terms of file systems?
- How is *hard-link* in typical Unix file system implemented?

File Systems - Aliases

- What is an *alias* in terms of file systems?
  - *Means of providing additional/alternate name for same file (i.e., to refer to blocks on disk associated with file from two different directory paths)*
- How is *hard-link* in typical Unix file system implemented?
  - *Add additional directory entry referring to same inode*

File Systems – Journaling

- What is journaling for file systems and why is it needed?

File Systems – Journaling

- What is journaling for file systems and why is it needed?
  - *Journaling is a means of ensuring integrity in a file system in the event of a failure (e.g., power failure) during modification to the file system*
  - *It is needed because typical disks guarantee atomicity of single block operations, but not multiple block operations. Many modifications to a file system require multi-block operations.*

File Systems – Blocks

- Describe one method of keeping track of free blocks in a file system
- What is the best block size to choose when formatting a partition with a file system?
- What are the performance tradeoffs in choosing the block size?

File Systems – Blocks

- Describe one method of keeping track of free blocks in a file system
  - *A linked list of free blocks (blocks of free blocks linked together)*
  - *A bitmap of free blocks (1 bit for each free block)*
- What is the best block size to choose when formatting a partition with a file system?
  - *It depends. For many small files, small blocks will mean less wasted space (internal fragmentation). But for larger files, large blocks can be more efficiently read and allocated.*
- What are the performance tradeoffs in choosing the block size?
  - *Larger block sizes generally has better maximum throughput, but smaller block sizes generally have better disk efficiency (less internal fragmentation).*

Sockets+

- What does `bind()` do?
  - Who calls `bind()`, the client or server?
- How do you re-direct stdout to a socket?

Sockets+

- What does `bind()` do?
  - Who calls `bind()`, the client or server?
  - *Bind assigns local protocol address ("name") to a socket. Bind is typically called by the server, to allow client to reach at well-known port (and address)*
- How do you re-direct stdout to a socket?
  - *`dup2(first, second)` – create a copy of a file descriptor (first to second), closing second as needed*

HLM02

- Compare and contrast WAFL inodes to traditional i-nodes.
- What is a snapshot?
- How is it implemented?
  - What is copy-on-write?

HLM02

- Compare and contrast WAFL inodes to traditional inodes
  - *Similar in that meta data (e.g., owner) and block pointers*
  - *Different in that WAFL pointers all same (e.g., all direct or all indirect) and really small files in inode*
- What is a snapshot?
  - *A "copy" of the file system at a given time*
- How is it implemented?
  - What is copy-on-write?
  - *Snapshots are implemented by copying the root inode. Any subsequent change to files copy data (including all blocks of meta data).*

HML02

- Performance methodology for NFS appliance?
- Why not simply time to `open() + read()`?
- Why not simply top throughput?

HML02

- Performance methodology for NFS appliance?
  - *Apply workload (e.g., LADDIS) to appliance, where workload produces range of NFS requests per minute*
  - *Look for "knee", where response time sharply increases*
- Why not simply time `open() + read()`?
  - *NFS servers may be fast for basic operations, but care about scalability as support many users and load*
- Why not simply max throughput?
  - *Users care about more than just max data rate, also care about how fast individual request provided – response time*

Distributed File Systems

- Compare and contrast *stateful* vs. *stateless* server for distributed file system

Distributed File Systems

- Compare and contrast *stateful* vs. *stateless* server for distributed file system
  - *Stateful (server maintains client states)*
    - *Shorter requests (since already authenticated, have last access)*
    - *Quicker request processing*
    - *Cache coherence possible*
    - *File locking possible*
  - *Stateless (server no info on clients)*
    - *Longer requests with access/offset*
    - *No open/close neede*
    - *Easier for server to recover from crash*
    - *No server state for client → more scalable*
    - *Cache coherence problem*
    - *No file locking*

Distributed File Systems

- How does NFS (v3) handle potentially out-dated client caches?

Distributed File Systems

- How does NFS (v3) handle potentially out-dated client caches?
  - *Server stateless so client most poll*
  - *Client read:*
    - *~3 seconds for file, ~30 seconds for directory*
  - *Client write:*
    - *Send "dirty" block about every 30 seconds*

Distributed Systems

- What are 3 techniques to scale distributed systems? What are the issues with each?

Distributed Systems

- What are 3 techniques to scale distributed systems? What are the issues with each?
  - *Hiding latency – do server-type computations on client-side. Issue? Client capabilities, "cheating"*
  - *Distribution – spread information processing to more than one location. Issue? "Routing" to find information, performance*
  - *Replication – copy information to increase availability. Issue? Consistency*