**HDFS Details**

* Optimized for handling large files
  + Big logs from webservers, sensors, etc
* Breaks files up into blocks
  + 128MB by default
  + Not limited by limitation of a single hard drive
  + Broken out -> can be distributed across multiple storages
* Can distribute the processing of the entire file up since the original data is broken up into chunks
  + Optimize by having machines handle chunks of data closer to itself
* Blocks stored across several commodity computers
  + Horizontal scaling
  + When a single machine is down -> fail-over and go to another machine since each machine contains copy
* Name Node vs Data Node
  + Name Node
    - Keeps track of state of blocks
    - Knows where to go to retrieve a copy of a given block
    - Edit log
      * Tracks insert, modify, and delete
  + Data Node
    - Communicates with the client (requestees)
    - Physically stores data blocks
  + Flow
    - Read
      * Client -> (API) -> Name Node -> Client
        + Retrieves which data nodes store the information
      * Client -> Data Node
        + Physically fetches the data from data nodes
    - Write
      * Client -> (API) -> Namenode -> Client
        + Send request for data write
        + Name Node tracks the change
        + Tells the client to interact with a Data Node to physically store
      * Client -> Datanode
        + Request a physical write
      * Datanode -> Data Node (all other)
        + Replicates the data
      * Datanode -> Client
        + Returns Write Status
      * Client -> Namenode
        + Serves up the write status
        + Namenode updates
  + Name Node Resilience
    - Back up Metadata
      * Namenode writes to local disk and NFS
        + NFS -> Network File System
      * Use backup incase Namenode fails
    - Secondary Namenode
      * Maintains a merged copy of edit log
        + Restore from log
    - HDFS Federation
      * Each Namenode manages a specific “namespace”
      * Suitable for smaller data
      * Given data volume communicates with specific namenodes to retrieve data
    - HDFS High Availability
      * Hot standby namenode using a “shared edit log”
        + Shared edit log stored in a separate datastore (not HDFS?)
      * Zookeeper tracks active namenode
      * Forces only one namenode to operate at a given point in time
        + To keep data integrity

**Using HDFS**

* UI (Ambari)
* cmd line interface
* HTTP/HDFS proxies
  + Can setup a web interface
* Java Interface
  + Can program in other languages to talk to Java code
  + Most of Hadoop is written in Java
* NFS Gateway
  + A way of mounting a remote file system on a server
  + Mount HDFS on a linux box (or any system)