HdfsFutures

HDFS Futures

Below is a categorized list and descriptions of HDFS Future Features

Goal: HDFS for Production Use

1. Reliable and Secure: The file system is solid enough for user to feel comfortable to use in "production"
    - Availability and integrity of HDFS is good enough
        - Availability of NN and integrity of NN Data
        - Availability if the file data and its integrity
    - Secure
        - Access control - in 0.16
        - Secure authentication 0.19
2. Good Enough Performance: HDFS should not limit the scaling of the Grid and the utilization of the nodes in the Grid
    - Handle large number of files
    - Handle large number of clients
    - Low latency of HDFS operation - this will affect the utilization of the client nodes
    - High throughput of HDFS operations
3. Rich Enough FS Features for applications
    - e.g. append
    - e.g. good performance for random IO
4. Sufficient Operations and Management features to manage large 4K Cluster
    - Easy to configure, upgrade etc
    - BCP, snapshots, backups

Service Scaling

This means scaling the Name Service (aka Namenode) and the number of Datanodes that can be present in a HDFS system.

For scaling the Name service (Namenode), there are two main issues here

- scaling the name space (i.e. the number of files and directories we can handle
- scale the performance of the name service - i.e. its throughput and latency and in particular the number of concurrent clients

Improving one may improve the other.

- E.g. moving Block map functionality to slave NNs will free up storage for Name entries
- E.g. Paritioning name space can also improve performance of each NN slave

Summary of various options that scale name space and its performance (details below)

(Also see ScaleNN_Sea_of_Options.pdf)

- Grow memory
    - Scales name space but not performance
    - Issue: GC and Java scaling for large memories
- Read-only Replicas of NN
    - Scales performance but not namespace
    - Adds reliability and one of the steps towards HA
- Partition Namespace statically: Multiple namespace volumes
    - Scales both
    - Retains HDFS's design philosophy but need a simplified automounter and management of horizontal scaling of NN servers
- A truly distributed name server that automatically partitions the namespace dynamically
- Split function on NN (Namespace and Block maps)
    - scales name space x3, a little performance scaling
- Page-in partial name space from disk (as in traditional FSs)
    - Scales namespace but not performance, unless multiple volumes

Scaling Name Service Throughput and Response Time

- Distribute/Partition/Replicate the NN functionality across multiple computers
    - Read-only replicas of the name node
        - What is the ratio of Rs to Ws - get data from Simon
        - Note: RO replicas can be useful for the HA solution and for checkpoint rolling
    - Partition by function (also scales namespace and addressible storage space)
        - E.g. move block management and processing to slave NN.
        - E.g. move Replica management to slave NN
    - Partition by name space - ie different parts of the name space are handled by different NN (see below)
        - this helps in scaling the performance of NN and also the Name space scaling

- RPC and Timeout issues
  - When load spikes occur, the clients timeout and the spiral of death occurs
  - See Hadoop Protocol RPC
- Higher concurrency in Namespace access (more sophisticated Namespace locking)
  - This is probably an issue only on NN restart, not during normal operation
  - Improving concurrency is hard since it will require redesign and testing
    - Better to do this when NN is being redesigned for other reasons.
- Journaling and Sync
  - *Benefits*: improves latency, client utilization, less timeouts, greater throughput
  - Improve Remote syncs
    - Approach 1 - NVRM NFS file system - investigate this
    - Approach 2 - If flush on NFS pushes the data to the NFS server, this may be good eough if there is a local sync - investigate
  - Lazy syncs - need to investigate the benefit and cost (latency)
    - Delay the reply by a few milliseconds to take allow for more bunching of syncs
    - This increases the latency
  - NVRAM for journal
  - Async sysncs [No!!!]
    - reply as soon as memory is updated
    - This changes semantics
      - If it is good enough for Unix then isn't it good enough for HDFS?
        - For a single machine, its failure implies failure of client and fs *together*
        - In a distributed file system, there is partial failure; further more one expects HA'ed NN to not loose data
- Move more functionality to data node
  - Distributed replica creation - not simple
- Improve Block report processing HADOOP-2448
  2K nodes mean a block report every 3 sec.
  - Currently: Each DN sends Full BR are sent as array of longs every hour. Initial BR has random backoff (configurable)
  - Incremental and Event based B-reports - HADOOP-1079
    - E.g when disk is lost. or blocks are deleted, etc
    - DN can determine what if anything has changed and send only of there are changes
  - Send only checksums
    - NN recalculates the checksum, OR has rolling checksum
  - Make intial block report's random backoff to be dynamicaly set via NN when DNs register. - HADOOP-2444

Scaling Namespace (i.e. number of files/dirs)

Since the name node stores block and name objects in memory, the size of the name space (and hence the number of files) is limited by amount of heap memory. Currently a 14GB heap (ie 16GB machine) allows 60 million block and name objects. Hence if one has 2 blocks per file, then one is limited to 20 million files. This is a significant restriction for large clusters. Besides adding more memory, several options are listed below.

Partition/distribute Name node (will also help performance)

Several Options:

- Statically Partition the namespace hierarchically and mount the volumes
  - In this scheme, there are multiple namespace volumes in a cluster.
  - All the name space volumes share the physical block storage (i.e. One storage pool)
  - Optionally All namespaces (ie volumes) are mounted at top level using an automounter like approach
  - A namepace can be explicitly mounted on to a node in another namename (a la mount in Posix)
    - Note the Cepf file system [ref] partitions automatically and mounts the partition
- A truly distributed name service that partitions the namespace dynamically.
- Only keep part of the namespace in memory.
  - This like a tradional file system where the entire namepsace is stored in secondary and page-in as needed.
- Reduce accidental space growth - name space quotas

Name Service Availability (includes integrityof NN data, HA, etc)

Integrity of NN Image and Journal

- Handling of incomplete transactions in journal on startup
- Keep 2 generations of fsimage - checkpoint deamon is verifying the fsimage each time it creates the new one.
- CRC- for fsimage and journal
- Make the NN persistent data solid
  - add internal consistency counters - to detect bugs in our code
    - Num files, Num dirs, num blocks, sentenials between fields, strong lengths
- Recycling of block-Ids - problems if old data nodes come back - fix has been deisgned
- If failure in FSImage, recover from alternate fsimages
- Versioning of NN persistent data (use jute)
- Smart fsck
  - Bad entry in journal - ignore rest

    - Bad entry in journal - ignore only those remaining entry not effected (Hard)
    - If multiple journals, recover from the best one or merge the entries
    - NN has flag about whether to continue on such an error
- Recreating NN data from DN will require fundamental changes in design

Faster Startup

- Faster Block report processing (See above)
- Reload FS Image faster

Restart and Failover

- Automatic NN restart on NN failure (operations can add stander watchdog for this)
- Hot standby & failover

Security: Authorization and ACLs

- 0.16 has access control with very weak authorization
    - Client side grabs OS user id and passes it to NN
- Secure authorization 0.19
- Service-level access control - ie which user can access the HDFS service (as opposed ACLs for specific files)

File Features

- File data visible as flushed
    - Motivation: logging and tail -f
- Currently if an open files is renamed or deleted, the client with the open file can get an exception
    - We are unlikely to fix this as keeping a list of open files on NN side is probably too expensive.
- Growable Files
    - via atomic append with multiple writers
    - Via append with 1 writer Hadoop-1700
- Truncate files
    - Use case for this?
    - note truncate and append needs to be designed together
- Concatenate files
    - Here multiple files are concatenated by merging their block lists (ie not data is copied)
    - This will require support forvariable length block.
    - Reduces number of names but since # of blocks are same does not offer much name space scaling
- Support multiple writers for log file
    - Alternatives
      1 logging toolkit that adapts to Hadood
        - Logging is from within a single application
        - No changes needed to Hadoop
          1 atomic appends takes care of that - overkill for logging??
- Block view of files - a file is a list of block, can move block from one to another, have holes etc

File IO Performance

- In memory checksum caching (full or partial) on Datanodes (What is this Sameer?)
- Reduce CPU utilization on IO
    - Remove double buffering Hadoop-1702
    - Take advantage of send file
- Random access performance Hadoop-2736

Namespace Features

- Hardlinks (not realy needed)
    - Will need add file-ids to make this work
- Symbolic links
- Native mounts - mount Hadoop on Linux
- Mount Hadoop as NFS
- "Flat name" to file mapping
    - Idea is remove the notion of directories. This has the potential of helping us scale the NN
    - Main issue with this approach is that many of our applciations and users have a notion of a name space that they own. For example many mapReduce jobs process all files in a directory; another user creating files in that directory can mess up the application.
- Notion of a fileset - kind of like a directory, except that it cannot contain directories - under discussion - has potential to scale name node.

File Data Integrity (For NN see NN data integrity above)

- Periodic data verification

Operations and Management Features

- Improved Grid Configuration management
  - Put config in Zookeeper (would require that NN gets started with at least one ZooKeeper instance)
  - DNs can get most of their config from NN. The only DN specific config is the directories or the DN data blocks
- Software version upgrades and version rollback P???
  - See Rpc Protocol Versioning
- Rolling upgrade when we have HA

- NN data rollback
  - this depends on keeping old fsImage/journals around
  - A startup parameter for this?
  - Need an advisory on how much data will be discarded so that operator can make an intelligent decision
- Snapshots
  - We allow snapshots only when system is offline
  - Need live Snapshots
  - Subtree snapshots (rather then whole system)
- Replica Management
  - Ensure that (R-1) racks for R replicas
  - FSCK shold warn that there aren't R-1 racks

Hadoop Protocol RPC

RPC Timeouts, Connection handling, Q handling, threading

- When load spikes occur, the clients timeout and the spiral of death occurs
- Remove Timeout, Instead Ping to detect server failures HADOOP-2188
- Improve Connection handling, idle connections etc

Client-side recovery from NN restarts and fallovers

- HDFS client side (or just MapRed?) should be able to recover from NN restarts and failovers

Versioning

- Across clusters
  - Versioning of the Hadoop client protocol, server can deal with clients of different versions
    - The data types change frequently, fields added, deleted
- Within cluster - between NN and DN

Multiple Language Support

- Are all interfaces well defined/cleanup
- Generate stubs automatically for Java, C, Python
- Service IDL

Benchmarks and Performance Measurements

- Where are the cycles going for data and name nodes?
- For HDFS and Map-Reduce

Diagnosability

- NN - what do we need here - log analysis?
- DN - what do we need here?

Development Support

- What do we need here?

Intercluster Features

- HDFS supports access to remote grids/clusters through URIs

- Federation features - investigate
- What else?

BCP support

- Support for keeping data in sync across data-centers

Attachments

<<AttachList>>