the Hadoop distributed file system

wbg231

January 2023

1 Introduction

- hadoop is designed to store very large datasets relaibly and to stream those data sets at high bandwith to user applications
- the idea is to dsitribute storage and computation across many ervers allowing reaserouces to grow with demand and remain economical ie cheap lol
- was made at yahoo

Introduction and related work

- hadoop provides a distributed file system and a framework for computation using the map reduce paradigm
- hadoop works by partitioning data and computation across many hosts and keeping in parallel close to where the data is kept
- hadoop scales simply by just adding more servers
- components of hadoop
    1. hdfs the file system
    2. map reduce the frame work for distributed computation
    3. hbase column oriented table service
    4. pig data flow language with parallel execution
    5. hive data whorehouse infrastructure
    6. zoo keeper distributed coordination service
    7. arvo data serialization service
- hadoop is an apache project
- most of the hdfs interface is modeld after unix but some changes were made to improve preformence

- HDFS stores meta data on a dedicated server called the name node
- all servers are fully coneected and able to comunicate with another via TCP based protocols
- file is made durable by being replicated on multiple file nodes
- this also allows improvents to data transfer bandwith as it is more lickly that data will be located near the computer that needs it

architecture

name node

- the HDSFS name space is a hierachy of files, files and directories are represented on the name node by inodes chich record atributres liker permisions modifcations and acess times
- the file content is split into large blocks and each block of the file is replacated at multiple ussualy three data nodes.
- hte node names maintains the name space tree and mapping of file blocks to the data node(ie where the data is physcially stored)
- an hdfs client wanting to read a file first contact the name node to lcoate the data locks comprising the file and then reads the blocks conetnts form the nearest data node/ '
- when something is written to a file the name node nominated three data nodes to store the updated data.
- HDFS keeps the entire name space in ram.
- all teh meta data is called the image
- the back up of the image is called a checkpoint

data nodes

- each data blcok replica on a data node is represented by two files on the hosts native file system.
- the first file has the data hte second has the blocks meta data
- host computers can share multiple blocks so if one block only takes up as mcuh sapce as it can fill
- when starting a program the name node checks that each data node has the right name space id and software versions

- the namespace id is asigned to the file system instance when it i formatted, the name sapce id is persentaly sotre on all nodes of hte cluser, so nodes with diferent namespce id can not join a cluster keeping inegrity (ie no wierd computers on your stuff)
- the name node stores the storage id of each data node which registers which is a unique identfier.
- datanode idenfties block replicas to the name node by sending a block report
- the data node sends ping the name node regularly with heart beats if these do not come through for a period of time the data node is thought to be dedicated

hdfs client

- users work with hdfs using hte hdfs client
- user can write or read files. gives instructions to name node writes and reads from data nodes

image and journal

- the namespace image is hte file system meta data that describes the organization app data files and directories
- the journal is a commit log for changes to the file system
- the name node in addition to serving client requests can also work as either a checkpoint or backup node
- the checkpoint node combines the existing checkpoint and journal to make a new checkpoint (the checkpoint node typically run on a different host than the name node)
- having checkpoints is a good way to ensure system data will be ok in the event of a crash

backup node

- the backup node is a a read only name node that works kind of like a checkpoint node but stores the most recent journal in its local memory so if the name node goes down the information is quickly accessible

file and io operation

file read and write

- apps can add data to hdfs by creating a new file adn writing data to it. after the file is closed the it can not be written to except by append's that is adding stuff to the end
- when a client tries to write to a file they ping the name node and are the only system allowed to write to that file until there transaction is done
- when file is to large for current block, the name node makes new blocks and tells certain data nodes to hold replications
- if a user tries to read a file, it is conected to the nearest data node, if that read does not work it is conected to the data node with the nearest replica and so on
- a user can read te file from a data node that is being written to, but it gets the last line (ie where that file could be be modfied) from another replication of the file

block placement

- nodes are organized in racks.
- nodes of a rack share a switch
- racks are coneected by core switches
- comunication between two nodes in different wracks had to go through multiple switches
- the network bandwith between nodes in the same wrack is greater than that between wracks
- network bandwith between nodes is estimated by there distance
- when a new file is created hdfs places the first replica on the node where the writer is located. the second and third on two nodes in a diferent rack and the rest are places randomly to ensure that if any single node or wrack goes down that file is not lost

replication managment

- the name node tries to always ensure that each block has the correct number of replicas

- it priortizes adding replicas to blocks that have to few over deleting replicas from blocks that have to many
- always makes sure that not all replicas of a block are on a single node

balancer

- there is a balancer tool that tries to spred disk usage evenly across the hdfs cluster
- there is also a block scanner that periodaclly makes sure that the meta data of each block is in line with there block id
- the rest of this paper is about practices at yahoo