Big data in class final

wbg231

January 2023

Introduction

- ok they are getting started
- final is friday at 6pm
- need pencil adn id
- can bring 1 page of hand written notes
- don't bring any devices
- exam consists of 60 multiple choice questions which will be evenly distributed
- there are only conceptual questions, not code. this is stuff that could come up in an interview high level conceptual questions
- just hand write notes, lets just keep stuff simple aim to write less.
- going to be on a scan tron
- focus on things that are tome some extent at a high level
- around the same level of detail as the quiz

survey results

- there were not many survey results
- tried to make review slides follow that distributions
- most were focused near the 3rd quarter of the class.

relational databases

- not much to say
- they help standardize how data is
- data validation is also helpful
- speed up access
- sql is good
- ACID rules
- atomicity if have a sequence of operations either all complete toghter or all fail togther
- related to relational databases but not specfic to it, it is a thing for any concurent databases
- dutability is where we pass to the operating system
- consistency just means that everyhting will stay in line with the schema always in valid started
- independence order of execution is irrelevant this matters less because we have read only data in most stuff

map reduce

- map: process 1 record a time genearte key value pairs
- reduce: process 1 key at a time output single object (these are paralell)
- combiner: kind of like a reducer in the mapper
- constrianing form of computation to make things faster (really really tight constraints)
- combiner trys to minimize key skew, by doing a partail reduction
- combiners only really work if certain conditions are met

complexity analysis

- analyzing complexity
- just count the number of loops my man sheesh
- input of some length

- how many steps does that algorithm tkae to complete that results
- we only care about the dominating term and the lower bound $O(n)$
- can do teh same thing for space
- in the context of map reduce have to loop at the number ofkeys, number of reducers, number of mappers
- questions about combiner,

map vs reduce

- reducer always sees key and list of values for that key, mapper just sess imput keys
- reducer sees all the values for a key, mapper just sees one record at a time
- the mapper should be independence of one another

map reduce questions

- reducers do not reduce the number of keys, they reduce the number of values per key the number of jobs that goes to the reducer will be the same
- a map function in map reduce does not have to produce any intermediate value for every line, like if we were filtering for a word

HDFS

- name nodes vs data nodes
- data nodes store the data
- name nodes sotre the name
- in HDFS large file broken into blocks, the data nodes store the blocks but not the meta data saying where the data is from
- the name node maps file names to the data nodes that contian the block, the name node is your lookup table the data node is your storage
- in HDFS we get around concurent acess stuff, by just keeping the files read only
- we do not have to worries about t

- repelcation factor, defualt repelcation rate is 3. 1 copy on 1 mahcine, 1 copt on a difrent machine on teh same wrack, 2nd copy on a seprate wrack if that wrack goes down
- raising the repelcation factor means that it is going to be easier to find the data that you need at the cost of storage
- does it cost computation effeciceny to increase repelcation rate, no it is increasing data? it will increase storage cost obv tho
- data is append only, does that mean we can add info to the same file?, the idea is that you are allowed to append ie just change the last block in teh file once the block is full you create a new block.
- this is why when you run teh same map reduce program with out cleaning up output files it crashes?
- the mapper write back to temp files on hdfs that are cleaned up by map reduce,

questions

- if the name node fails you are in trouble.
- it happens it is not permanent it just means there is server down time
- this goes back to the CAP therome, we are giving up A when the name node fails, you need to wait for the server to not be crashing

spark

- the bottom line is that map reduce is form mid 2k it was good for what it was meant to do but not all tasks
- but we wanted to do other tasks, like iterative tasks,
- in map reduce all the reducers sit around idely until teh mappers finish that is not good if we are doing iterative sutff
- spark uses RDD where we sepeate actions form transfomrations. we can do a lot of transfomrations this allows for efficent computaiton
- wide vs narrow dependicy a narrow dependicy is easy to parallelize. the solution is almost always make sure that if you are joining RDD's they have teh same partion sturcutre (and thus are a narrow dependicy) the partions can map directly, the output depends on at most one input partitin so can propigate the same partion sturcutre

- a wide partion is something that dependds on all our data data, joins are wide unless two tables have same partion strucutres
- want dependicy narrow or it is hard to parallelize
- spark is a better implementation for SQL as well as machine learning
- SPARK gets the same parallelism as map reduce with less ristirvtions
- spark is a direct response to stone breaker
- relation between spark and map reduce, spark is seperate from map reduce it is a completly seperate thing
- ...

spark questions

- compare adn contrast rdd and adta frame
- spark data frames are not reaed only that i the whole point that is why we have ittertive stuff going on.
- each step in an rdd lineage graph must be compltetd fbefore th enext false
- spark uses pieples to cncet mutliple stages of map reduce proessing. false no it is a sepeate thing that was made to adress issues from map reduce
- spark can use hadoop but hadoop and map reduce are not the same thing
- are teh action going to be sequential,
- lineage graph is just about transfomrations, the actions are outside of the rdd
- rdd's move backwards though the computaiton graph until it finds a data source

coloumn oriented storage

review

- the defualt for spark is parquet
- this makes things faster sometimes jsut need 1 coloumn but not all rows.
- regularity helps with that this leads to speed ups
- column oriented storage becuase col have same data types all us to search through them faster

- can use dremel that alays truns a hierchal object into a tabular rerpreation that we can view as a col oriented storage
- the bigger take away is that dremel was cool but no one uses it
- parquet is actually used.
- csv files are a lot slower than parquet
- this is why the project data was all parquet not csv
- parquet is not human readbale tho

questions

- explain parquet and dremel, just udnerstnd the relationship between the two .
- the dremel system was designed to process all subsets of records in a data set. n it is for sunsets of atriibures for all records. teh point is it is not record based it is feature based
- when written to hdfs parquest files localte difrent cols in difrent hdfs blocks that is false, parquet files devide blocks by rows not col
- what is a block is a subset of records a contiogous chunk of a dataframe but spanning all the coloumns, that is why the whole thing is called parquet

dask

review

- there were a lot of dask thoughts
- hwen would you want to use dask?
- it is kinda dependt, spark is good if you are ok doing everyhting in spark if you need to do things where spark interacts with something in pyhton that is not written in spark, than it gets a bit messy to put things in and out of spark. so dask is ncie for that kinda thing
- most of the time you will be dealing with sql
- after that spark number 2, spark is good for regular data
- for ragged data or messy data dask can be usefull, or if you need parallelism that is not rows in a data frame but chunks in an array, get more flexability in your data types

- scalal makes constrains makes stuff faster
- dask accepts what ever comes out of your pyhton code, but hard to figure out waht you are doing
- you have to work harder to make dask work well
- but there are some jobs where dask is ideal
- daks is for data scicne use cases not software engeneering stuff

questions

- cross compare spark and dask. dask is spark written by scipy poeple, the computational princples are similar. in terms of how to optimize stuff it comes back down to partiion sturcutre need to minimze comuncaiton early on as well
- in some programs just have to loop, the thing is you need to identify when the task is worht it.
- no strong words of advice
- in sql indecies are how you optimze those should be designed for the queries you think you want to run
- those are just additional datastrucutres that can make things faster but do not effect ur data

aproximate nearest neightbors

- this is all similarty search
- one of the key aplications is sim saerch
- that is a fundemtnal use case
- as you increawse block size in lsh your liklyhood of colision goes down but have to do more work, so there a trade off with block size and number of blocks
- similarty search lsh lets you do this in a really fast way
- boost the high similarty item reduce the low similarty item
- brute force similarty is overwhelming for large documnets that is the high lvel thing.
- we did not really talk about distance metrics

questions

- lsh is built on the idea of randomness in mini has that is from the hash function but that can be wastefull saptial tree is instead of doing indepednint paritons we do resursive splitting, we are cut our data point in half as many times as we want. this is effecticly many locally depdnt hash functions
- cosine similarty lsh if you want to aproximate cosine similart can check the lsh by randomly projecting them
- multi probe not requiring that stict a match for all values
- multiprobe at every sate fuz the results a bit
- at the high level cosine similarty is very genearl
- min has failes when a single elment belong to every set in a colection true
- min has sifgureates are genearted by aplying false the min has functions has not direclt ofmared in min hash

reproducability

- this is really important in big data
- it is hard to reproduce results in big data and they propigate over large dataset
- reproducability can you reproruce this.
- most of the results from the 90's can not be reporduced
- there are all kinds of best practices that help with reproducability
- want to have three coppies of all your data files, at least 1 backup off site?
- want to have organzied project strucutre, input data, process code, results results, metadata
- want to have at least one readme file on how to use things
- in jupyter note books some times cells are not even run in orderd
- there are safer ways to store sensative data
- verion contorl use git
- that is super important in big data
- get the same result so can retrace your steps

recomender systems

- idea is to predcit whcih imtes a user wil interact with form a large catalog
- what is the dampening facotr? it is how we considerd items with more interactions to be a better esitmate of itnerctions adn make sure the wheigh more
- the next step is the latenent factors model
- decompose the model into seperate vectors
- implicit vs explicit feedback, explicit liek radings implicit like if you watch a youtube vdio or waht ever
- explicit feedback is ussualy a storng signal that is a clear indecator but is often of a lower volume
- implicit feedback tends to higher volume with lower signal to noise feedback,

saerching ranking evaluation

- google used ranom surfer
- page rank etc
- ranked lists evaluations for seach recomendaiton systems etc
- the thing that we say is that this cooks down to a linear algebra problem
- rank list evaluation is important
- the simplest model is that we make a predicted list of items for eaech users, we want all the pluessed to come before all the minues. the detials between how you weight postives and negatives and where they are in teh list distingushed metircs, but the main point is evaluating a rank list as if it is a prediction

question

- in page rank, the model is random if that browser lands on a page that has no out going links of links to its self, then teh model can not get out of there
- so ig you ever have a change of ladning there, it will think it is important, so they introduce this teleprotiaton aprameter that lets your suffer telport to somehweere ranodm

difrential privaise

- ways to keep data more private
- in one sentance it is shown taht jut annoamity is nto sufficent
- guess who you can reduce earch sapce ina lot of dimenions
- once the thing is de anonoized it is out there :/ if that is sensative we have problems
- difrential privacy keeps raw data private for ever adn you interact with the dataset through an api
- add some specfic laplacian noise that can not easily be removed it is absolute value of expotnetial so hard ot remove
- it is harfd ot know if you are in teh data set if it is given
- for a larger dataset there is a lot of pluasable denialbility
- this is limited to multiple queries
- so privacy laws agreate
- the noise that we inject is at the level of the computaiton we ask in result for it is on a per queries basis not, in teh dataset it's self
- the raw data is unefected which is ncie :)
- but we inject noise on the level of the query so it allows for pluasable denialbility, the user of the api may know what type of noise was put in the dataset

gpu

- computational parallelism from dedicated hardware
- simialr to traditional cpu but with more resritive progrma flow
- same components jsut in difrent proprtions
- gpu many small processrs that do a few computaitons in small steps
- think of aech core on teh gpu as youching a difernt part of memory that is a lot of the wya there
- cuda frame work is thread ie indivdual expdlubtio goruped into blcosk whicha re groupe dinto grids
- adn that is how we devied the work up in a way that we can devide over out interite dataset we just run on one grid at a time until we are done

- can return arbitary data types
- a kuda kernel does not prevent you form looking at difrent data
- that is nice becuase you can do convoltions easily
- this allows limited comuincation ebtween threads but doing this effecticly requires some involved profmraing
- they are statnig to creep more into like normal data science aplications
-