MapReduce: A major step backwards

wbg231

January 2023

1 Introduction

- the authors are happy that map reduce has people talking about DBSM, but also has the following gripes
    1. it is a step backwards in the paradigm of large scale data intensive applications
    2. it is a bad implementation of the code
    3. it is nt novel and most of the techniques are 25 years old
    4. it is missing key features of modern DBMS
    5. it is incompatible with DBSM tools

a step backwards

- the data base community has learned the following over the past 40 (now like 60) years
    1. schemas are good
    2. the schema can not be modified by the application
    3. high level access languages are good (that is SQL is faster and easier than lower level languages like C++ when working with data)
- map reduce does not have schemas so there is now way to ensure that the data is correct or constituent so database systems can break
- as schemas are not kept a programer must find the schema for there data by inspection as they work. this is dangerous
- during the 1960's there was a debate between the relational model where (you use a querying language for database access) abd the codasyl view where you write an algorithm each time
- the results of this debate are clear, and map reduce is on the low level side so that is bad and tedious

- by working with key value pairs a map reduce function is in effect acknowledging the presence of a key and attribute and thus could be put in schemas

map reduce is a poor implementation

- modern DBSMSs have the ability to build indices which while not useful for all problems are much faster thn brute force when they work map reduce just uses brute force
- map reduce also has issues with skew and data interchange
- skew (or key skew) occurs during the map phase of map reduce when there is a wide variance in the distribution of records with the same keys. this causes some reduce calls to take much longer than others, meaning that the program run time ends up being held back by the slowest reduce instance (there are ways around this that map reduce could use)
- in map reduce each of the N map workers produce M output files where m is the number of partitions so that is overall n*M files which is a lot.
- then at the reduce phase each of the M reducers needs to read N files, this means there is a lot of comunication and when two reducers try to read form the same mapper at the same time things slow down a lot

map reduce is not novel

- this is just history not super relevent

map reduce is missing features

- missing bulk loaders that can transform input data files into a desired format adn load them into a DBMS
- missing indexing
- missing transactions (that is blocks of queries)
- missing updates to change the data in the database
- missing integrity constraints
- missing views (ie temporary tables that are not stored but used for computation and not saved to modified the data)

map reduce is incompatible with DBMS tools

- a modern DBSM can use all the following
  1. report writers to make visualizations
  2. Business intelligence tools - to allow for ad-goc queries from large data warehouses
  3. data mining tools to allow users to discover structure in large datasets
  4. replication tools to allow users to replicate data from another DBSM
  5. database design tools to help with database construction
- the lack of these tools makes map reduce hard to use for an end to end task