Def Hapoop

/

“In pioneer days they used oxen for heavy pulling, and when one ox couldn’t budge a log, they didn’t try to grow a larger ox. We shouldn’t be trying for bigger computers, but for more systems of computers.” – Grace Hopper

/

How does Hadoop abstract the problem of disk reads and writes?

-

By transforming into a computation over sets of keys and values.

/

Are Hadoop and Relational Database Management Systems (RDBMS) complimentary?

What is the difference?

-

Yes.

RDBMS are good for point queries and updates as the data has been indexed into a B-Tree.

Hadoop is good for ad hoc analysis on the whole dataset, where Sort/Merge is used in essence to rebuild the entire database.

/

What type of data does MapReduce work well on that an RDBMS does not?

-

semi-structured or unstructured (no predefined schema)

/

Why does data normalization pose a problem for MapReduce?

-

Because reading a record becomes a non-local operation

/

What two functions does a programmer write for a MapReduce function?

What do they do?

-

Map & Reduce, which define a mapping from one set of key-value pairs to another.

/

What is MPI?

Where does MPI excel? Where not?

-

Message Passing Interface, where multiple computers share a stored memory.

It works well for compute intensive tasks, but not for problems that need to access large volumes of data, since the movement of data to the computers from storage becomes the bottleneck.

/

How does MapReduce solve the data bottleneck?

-

Through data locality, a feature where the compute task is moved to the data, so computations are performed as close to the data as possible – computer, rack, data center.

/

How does MapReduce spare the programming from having to think about failure?

-

Map or reduce tasks that fail are automatically rescheduled by the system.

/

Is MapReduce a shared-nothing architecture?

-

Yes and no. Yes that no data is shared between mappers or between reducers, but data is passed from mappers to reducers.

/

Does Hadoop like large files or small?

What is a good default size to target?

-

Generally, it is more efficient to process a smaller number of large files. If the size is too small, managing splits and map task creation begin to dominate total job execution time.

64 MB is a good target.

/

In Hadoop, what are the two types of nodes that control job execution?

-

jobtrackers – coordinate all jobs and schedule tasks.

tasktrackers – run tasks and report progress to jobtrackers

/

Where do map task write their output?

-

To local disk.

/

What governs the number of map tasks?

The number of reduce tasks?

-

The number of mappers is governed by the number of splits.

The number of reducers is specified by the programmer.

/

With multiple reducers, how do mappers determine which reducer to send data to?

-

Via the partitioning function, which can be specified by the programmer, although the default hash function works very well.

/

What is the shuffle?

-

The dataflow from the map to reduce tasks. Tuning it can have a big impact on job execution time.

/

How can you minimize the data flowing through the shuffle?

-

By using a combiner function.

/

What properties must a combiner function have?

-

Commutative and Associative

/

What allows programming languages other than Java to be used with Hadoop?

-

Hadoop streaming, based on Unix standard streams.

/

How does streaming differ from the standard Java API?

-

Streaming is naturally suited for text processing, where it can easily read multiple lines at a time, while the Java API reads one record at a time (although close() can be used to check for EOF.)

/

What is HDFS?

-

Hadoop Distributed File System, the file system which comes with Hadoop, although other filesystems can be used.

/

What three things are HDFS designed for?

-

1) Very large files – 100s of MBs, GBs, or TBs

2) Streaming data access – write once, read many times, where the time to read the whole dataset is more important than reading the first record.

3) Commodity hardware – chance of node failure is high.

/

What three things is HDFS not a good fit for?

-

1) Low-latency data access – since HDFS is designed for high throughput of data (HBase is a better choice for low-latency)

2) Lots of small files – since the namenode holds the file system in memory, it limits the amount of files.

3) Multiple writers, arbitrary file modifications – only one writer is allowed and writes are always to the end of a file.

/

What is the standard file block size?

For HDFS?

-

512 Bytes

64 MB

/

Why are HDFS blocks so large?

-

To minimize disk seek time in regards to disk transfer time.

/

What are the two types of nodes in HDFS?

-

namenodes (the master) – manages the filesystem

datanodes (workers)

/

What is a secondary namenode?

-

A backup of the namenode.

/

What is an HDFS Federation?

-

A way to expand the filesystem namespace beyond the limit of what the namenode can store in memory. In a Federation multiple namenodes are responsible for different parts, i.e. /user & /share.

/

What files system interface is HDFS loosely based upon?

-

POSIX – with similar commands, permissions, etc.

/

What can help speed up data transfer?

What is the tradeoff?

-

Compression.

Space/Time is the tradeoff – how long the compression algorithm takes and how much space is saved. Compression can be used both to store the data and for mapper output.

/

What is Hadoop’s serialization format?

-

Writables

/

In Hadoop what allows Writables to be ordered in the shuffle phase without deserializing them?

-

Comparators

/

What is the limitation of Writables?

What is designed to overcome this?

-

Specific to Java.

Avro is designed to be language agnostic and is based on JSON.

/

What two programming tasks are a challenge in distributed systems?

-

Debugging and Profiling

/

How many output files are created?

What are they called?

-

One for each reducers.

part-00000, part-00001, …, part-n

/

What are the 4 types of Hadoop logs?

-

system daemon – for admin

HDFS audit – all HDFS requests, for admin

Job history – events such as task completion

Task – for each task tracker child process, location defined by HADOOP\_LOG\_DIR variable.

/

What are six things to think about for tuning?

-

1) Number of mappers – should run about a minute, combine files if too many

2) Number of reducers – should be slightly less than the number of reduce slots so can all finish in one wave.

3) Use a combiner – if able

4) Intermediate Compression – map output compression is almost always a benefit.

5) Custom serialization

6) Shuffle tweaks – shuffle has multiple parameters for tuning

/

Why is local profiling not really relevant for overall profiling?

-

Because most jobs are I/O bound, not CPU bound.

/

What is a possible problem that can reproduced only on the cluster?

-

Excessive memory usage.

/

How do you generally model complexity in MapReduce jobs.

-

By having more MapReduce jobs vice have more complexity in a single job. Either represented by a linear chain of jobs or an Directed Acyclic Graph (DAG).

/

What are higher level languages build on MapReduce that can help manage complexity?

-

Pig, Hive, Cascading, Cascalog, Crunch…

/

What is Oozie?

-

A system for running workflows of dependent jobs.

/

What steps constitute the job submission process (in MapReduce1)?

-

1) Ask jobtracker for a new id.

2) Check output specification (if output directory already exists)

3) Compute input splits

4) Copy needed resources to jobtracker directory

5) Tell jobtracker job is ready for execution

/

What does MapReduce guarantee about the input to a reducer?

-

That it is sorted by key.

/

Why do you want map and reduce functions to use as little memory as possible?

-

So the shuffle has as much memory as possible.

/

The general form of map and reduce functions? Combiner?

-

map: (K1, V1) -> list(K2, V2)

reduce: (K2, list(V2)) -> list(K3, V3)

combine: (K2, list(V2)) -> list(K2, V2)

/

What determines the partition?

-

K2 and V2 (in practice only the key)

/

Does streaming pass the key to the mapper?

-

No, just the value.

/

What is the default separation character for streaming?

Can it be changed?

-

Tab character

Yes.

/

What is a split?

-

A chunk of input to be processed by a single mapper?

/

What constitutes a split?

-

Records – key-value pairs

/

Do splits and records require files?

-

No, they are logical abstractions.

/

What is a secondary sort?

Is it possible to do one in streaming?

-

MapReduce sorts each record by key prior to passing to the reducer, but for any particular key, the values are not sorted. A secondary sort is the use of a composite key, where one part determines the partition (which reducer to go to) and the part determines the sort on that reducer.

Yes, with special classes such as mapred.output.key.comparator.