VE472 Midterm Review

Introduction to Hadoop

1. **What is the major issue of computer when facing big data? CPU or Memory or Throughput?**

   Throughput (data transfer rate), Von Neumann bottleneck.

2. **Ways of increasing throughput:**

   - Caching
   - Branch prediction
   - Parallel Read from multiple locations (RAID)

3. **How big is Big data? Or consider how large the data should be so that using Hadoop would be more efficient?**

   >1 TB. Generally petabytes in size. Hadoop clusters usually holds TB to PB of data and can process petabytes of data within minutes.

4. **History of Hadoop: when did it start? Which company developed Hadoop?**

   2002 as Apache Nutch. Apache Software Foundation.

5. **Goal of Hadoop:**

   Efficiently analyse massive amount of data.

6. **Basic components of Hadoop:**

   - common libraries
   - HDFS
   - MapReduce
   - YARN

7. **Why Java as programming language?**

   It is cross-platform.

8. **Pros and Cons of HDFS?**

   - High throughput
   - Large latency; metadata kept in namenode's memory; write always in append mode by a single writer.

9. **What is a container?**

   A container is an environment with restricted resources where application-specific processes are run

10. **What types of daemon does Yarn provide?**

    Resource Manager and Node Manager.

11. **What is YARN?**

    Resource manager/scheduler.

    - Interacts with the filesystem
    - Hides low level details from the user
    - Offers an intermediate layer supporting many other distributed programming paradigms

12. **What is the goal of Mesos, Myriad, Spark and Drill?**

    - Mesos: global scalable resource manager, not restricted to Hadoop
    - Myriad: use Mesos to manage YARN resource requests
    - Spark: Fully replace MapReduce; Support multi-pass applications; Write and read from the disk as little as possible; Take advantage of the memory
    - Drill: Integrate into Hadoop as a MapReduce replacement; Be an interactive ad-hoc analysis system for read-only data; Be easily expandable using storage plugins; Enjoy data agility

13. **Difference between Drill and Spark?**

    Drill: SQL query engine for Big Data exploration.

    - Drill allows fine grained security at the file level
    - SQL queries, searching -> use drill

    Spark: fast and general-purpose cluster computing system.

    - Spark can also do SQL queries.
    - Complex algorithms, ML & AI -> use spark

14. **List some other tools introduced in Hadoop ecosystem?**

    - Flink, Tez, Hbase, Hive, Spark SQL
    - Serialization and storage components
    - Management and monitoring (**Zookeeper**)
    - Analytics helpers

15. **Three layers of Lambda Data Architecture:**

    - Batch layer, storing data in batch
    - Speed layer, analyse the data
    - Serving layer, serve curated data

data are provided to batch layer and speed layer simultaneously.

Kappa Data Architecture: batch layer is removed.

16. **Difference between Batch processing and real-time processing?**

    - In batch processing, data is processed in parts. The data is first stored, and then processed. (Apache Hadoop - MapReduce)
    - In real-time processing, data is processed as soon as data is received, needs to be responsive and active. (Apache Storm, Apache Kafka, Redis)

Hadoop's Core Components

HDFS

1. **What is LVM?**

   Logical Volume Manager: manage disk partition.

2. **Default block size of HDFS**

   128MB

3. **Commands retrieving information of file**

   `lsattr`, `ls -l`...

4. **Pros and cons of having large/small blocks?**

   - Large block: good when dealing with large data, have low latency; bad since it may waste memory
   - Small block: save memory for smaller files, but wastes memory keeping track of free blocks, time-consuming when fetching data

5. **Jobs of namenode and datanode:**

   - Namenode is read only, maintains metadata of data in datanode, stores info in namespace image and edit log to locate datanode
   - Datanode store only the data or certain blocks in cache, reports the stored blocks to namenode

6. **What to do if the namenode fails?**

   Use backup namenode, via Network FileSystem(NFS) or `rsync`.

7. **When I have 2 namenodes, is it good or bad to have each namenode store half of the data nodes?**

   Bad because if one namenode fail, half of the datanodes are lost

8. **Having two namenodes in Active-Passive mode, when may race condition happen, how to avoid race conditions?**

   Active node goes down -> use passive node to write -> active node comes back -> have two active nodes writing -> race conditions.

   STONITH: shoot the other node in the head. If one namenode become active, kill the other node

9. **Default replication level of HDFS**:

   3:

   1. First: same node as the client.
   2. Second: random, different rack from the first
   3. Third: same rack as the second but different node
   4. Others: random node in the cluster

10. **Where should computation be done?**

    On rack holding second and third replication: data transfer on the same rack is fast.

11. **How does Distributed filesystem contact NameNode?**

    Via RPC(Remote Procedure Call) Connection

12. **How to handel failing in file write?**

    7 steps.

    1. Close pipeline
    2. Add packets
    3. Inform namenode
    4. Remove faulty data node
    5. Construct pipeline
    6. Complete writing
    7. Arrange replication.

YARN

1. **What is client node?**

   Client node is CPU (for calculation), in contrast data node is hard disk (for storage).

2. **Jobs of resource manager and node manager in YARN.**

   - Resource manager: Manage the nodes
   - Node manager: Start container that runs applications

3. **What is application master?**

   Application Master is a process that coordinates the execution of an application in the cluster. It is responsible for the execution of a single application. It asks for containers from the Resource Manager and executes specific programs (tasks) on the obtained containers. It is typically launched by Resource Manager and run in a container.

4. **Why are node manager nodes connected through subthreads instead of connecting directly to resource manager node?**

   Minimize traffic, decrease bandwidth, make things faster.

5. **Preferred location of the containers?**

   We aim to minimize data transfer time.

   - Best: the same as the node where data is stored,
   - OK: the computer on the same rack

6. **Three ways YARN are used:**

   - One application per user job
   - One application per user session
   - Long-running application shared among users

   No need to kill container for the last two case -> save time with previous data.

7. **Three schedulers in YARN:**

   - FIFO
   - Capacity (DEFAULT scheduler, waste resources, containers not killed inside a queue)
   - Fair (resource fairly shared, high latency due to allocation and deallocation of resources for different jobs)

8. **How does YARN solve the problem that an application requesting a busy node?**

   Each nodes send out heartbeat reporting the running containers and available resources. Capacity scheduler wait for some heartbeat before loosing the requirement. Fair scheduler wait for a predefined portion of nodes in the cluster to offer opportunities before loosening the requirement.

MapReduce

1. **Three steps of MapReduce:**

   - Map
   - Shuffle
   - Reduce

2. **Process of MapReduce job initializatin and startup?**

   5 steps:

   1. Request ID to RMCheck parameters
   2. Split into tasks
   3. Copy splits onto FS
   4. Submit job on RM

   7 steps.

   1. YARN allocates container
   2. RM launch app master
   3. Setup task
   4. Retrieve splits from FS
   5. Create Map tasks
   6. Allocate resources
   7. Locate data on FS

3. **Pros and Cons of MapReduce reading from/writing to disk?**

   Safe but slow. In contrast Spark and Drill will minimize disk usage.

4. **What to do when task fails, JVM crashes, or task hangs.**

   - *Task fails*: When receiving a failure notice the application master marks the task as failed. The container is freed and resources released
   - *JVM crashes*: the node manager notices the application manager of the failure
   - *Task hangs*: Tasks marked as failed if no report is received. The JVM is killed by the application master.

5. **Why are data compressed before sending to other reducers?**

   To reduce traffic. For the similar reason, only smaller part of data are sent to other reducers.

Drill, Spark and more

Drill

1. **Functions of Zookeeper:**

   Dependent of Drill. No large data-store, allow different nodes in cluster to communicate; let various applications of hadoop to work together.

2. **What is a drillbit?**

   A drill process created when running Drill on YARN. Each query given to drill is split into fragments and they are run on different drillbits.

3. **What is a foreman drillbit?**

   A drillbit that receives the query and drives the entire query. Every drillbit can be foreman drillbit.

4. **Characteristics of Drill:**

   - Each drillbit contains all services and capabilities of Drill
   - Columnar execution
   - Optimistic query execution
   - Vectorization
   - Runtime Compilation

Spark

1. **Two modes of spark:**

   - Client mode (Interaction with user)
   - Cluster mode (Runs on cluster, fully utilize hadoop)

2. **What is an RDD?**

   Resilient Distributed Dataset, a fundamental data structure of spark. It is a *read-only*, *partitioned* collection of records.

3. **What does `resilient` mean in RDD?**

   It is able to be reconstructed in case of partition loss, since it is storing how it was derived from other datasets.

4. **What does `distributed` mean in RDD?**

   RDD's elements can be partitioned across machines based on a key in each record..

5. **What make RDDs fast?**

   - Distributed collections of objects that can be *cached in memory* (drop using LRU) across cluster nodes
   - Manipulated through various parallel operations
   - Automatically rebuilt on failure.

6. **Two types of operations on an RDD:**

   - *Transformation*: creating new dataset, lazy evaluation (not executed until it sees an action, reorganize and optimize the process), avoid returning large datasets
   - *Action*: Compute on a dataset

7. **Caching levels of RDDs:**

   - Memory only
   - Memory and disk (RDD too long to reconstruct)
   - Memory only serialized (use Snappy to compress)
   - Replication (for security)

8. **Difference between Spark and MapReduce?**

   - Data storage: MapReduce use disk and play safe while Spark minimize disk usage, use memory as much as possible.
   - Upon failure: MapReduce use replication data; Spark simply reconstruct RDD
   - Speed: Spark is much faster

More tools

1. **What is Kubernetes?**

   A container orchestration tool, usually handles clusters that runs dockers.

Some Other Advice

1. Get familiar with how YARN, MapReduce, Drill and Spark work by looking at the flow charts on the slides,
2. Have a good review of your work in labs and homework: how clusters are set, how MapReduce is done, etc.