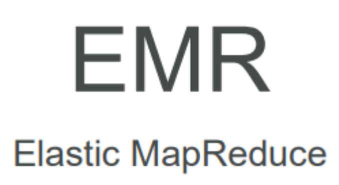
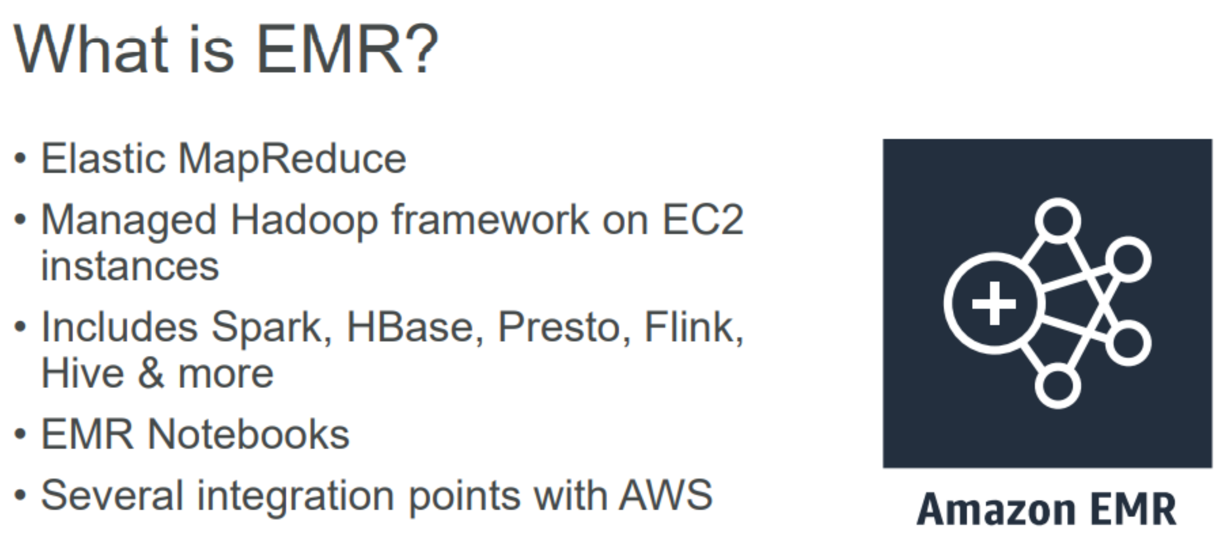
EMR Slides

09:56 PM



EMR

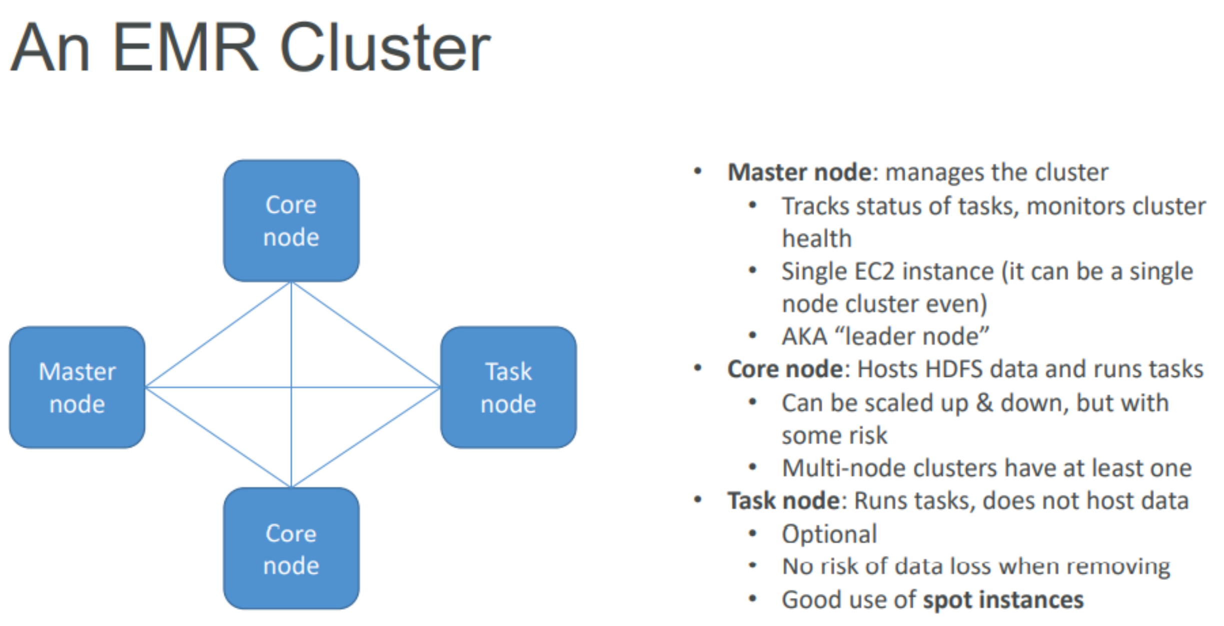
Elastic MapReduce



What is EMR?

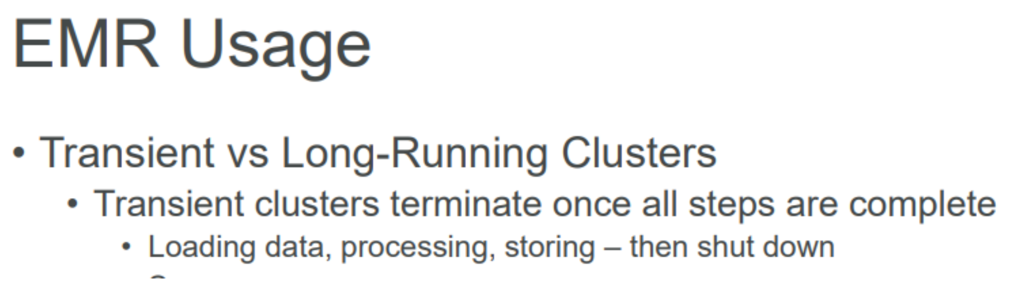
- Elastic MapReduce
- Managed Hadoop framework on EC2 instances
- Includes Spark, HBase, Presto, Flink, Hive & more
- EMR Notebooks
- Several integration points with AWS

Amazon EMR



An EMR Cluster

- **Master node**: manages the cluster
  - Tracks status of tasks, monitors cluster health
  - Single EC2 instance (it can be a single node cluster even)
  - AKA "leader node"
- **Core node**: Hosts HDFS data and runs tasks
  - Can be scaled up & down, but with some risk
  - Multi-node clusters have at least one
- **Task node**: Runs tasks, does not host data
  - Optional
  - No risk of data loss when removing
  - Good use of **spot instances**



EMR Usage

- Transient vs Long-Running Clusters
  - Transient clusters terminate once all steps are complete
    - Loading data, processing, storing – then shut down

- Saves money
- Long-running clusters must be manually terminated
  - Basically a data warehouse with periodic processing on large datasets
  - Can spin up task nodes using Spot instances for temporary capacity
  - Can use reserved instances on long-running clusters to save $
  - Termination protection on by default, auto-termination off

EMR Usage

- Frameworks and applications are specified at cluster launch
- Connect directly to master to run jobs directly
- Or, submit ordered steps via the console
  - Process data in S3 or HDFS
  - Output data to S3 or somewhere
  - Once defined, steps can be invoked via the console

EMR / AWS Integration

- Amazon EC2 for the instances that comprise the nodes in the cluster
- Amazon VPC to configure the virtual network in which you launch your instances
- Amazon S3 to store input and output data
- Amazon CloudWatch to monitor cluster performance and configure alarms
- AWS IAM to configure permissions
- AWS CloudTrail to audit requests made to the service
- AWS Data Pipeline to schedule and start your clusters

EMR Storage

- HDFS
  - Hadoop Distributed File System
  - Multiple copies stored across cluster instances for redundancy
  - Files stored as blocks (128MB default size)
  - Ephemeral – HDFS data is lost when cluster is terminated!
  - But, useful for caching intermediate results or workloads with significant random I/O
  - Hadoop tries to process data where it is stored on HDFS

- EMRFS: access S3 as if it were HDFS
  - Allows persistent storage after cluster termination
  - **EMRFS Consistent View** – Optional for S3 consistency
    - Uses DynamoDB to track consistency
    - May need to tinker with read/write capacity on DynamoDB
  - **New in 2021: S3 is Now Strongly Consistent!**

EMR Storage

- Local file system
  - Suitable only for temporary data (buffers, caches, etc)
- EBS for HDFS
  - Allows use of EMR on EBS-only types (M4, C4)
  - Deleted when cluster is terminated
  - EBS volumes can only be attached when launching a cluster
  - If you manually detach an EBS volume, EMR treats that as a failure and replaces it

EMR promises

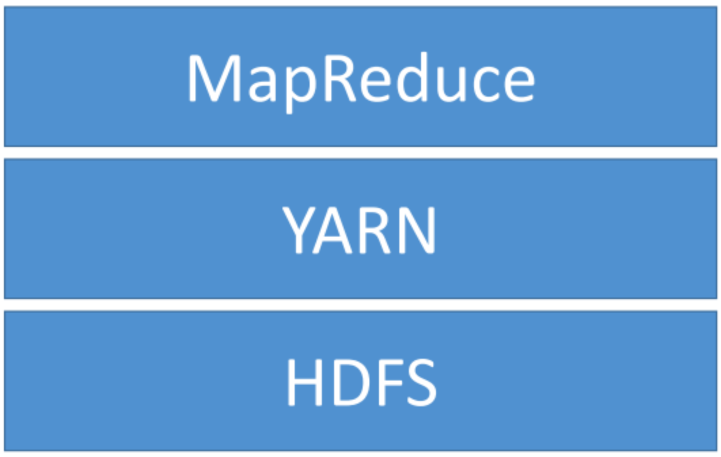
- EMR charges by the hour
  - Plus EC2 charges
- Provisions new nodes if a core node fails
- Can add and remove tasks nodes on the fly
  - Increase processing capacity, but not HDFS capacity
- Can resize a running cluster's core nodes
  - Increases both processing and HDFS capacity
- Core nodes can also be added or removed
  - But removing risks data loss.

EMR Managed Scaling

- EMR Automatic Scaling
  - The old way of doing it
  - Custom scaling rules based on CloudWatch metrics
  - Supports instance groups only
- EMR Managed Scaling
  - Introduced in 2020
  - Support instance groups and instance fleets
  - Scales spot, on-demand, and instances in a Savings Plan within the same cluster
  - Available for Spark, Hive, YARN workloads
- Scale-up Strategy
  - First adds core nodes, then task nodes, up to max units

  - First adds core nodes, then task nodes, up to max units specified
- Scale-down Strategy
  - First removes task nodes, then core nodes, no further than minimum constraints
  - Spot nodes always removed before on-demand instances

So… what's Hadoop?

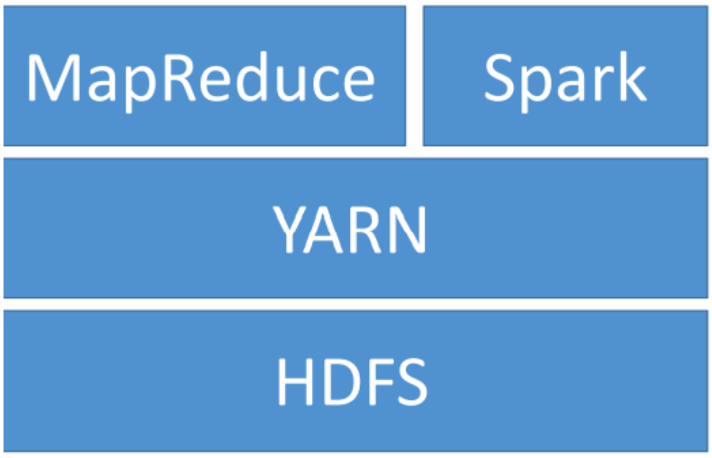


Framework for distributed data processing
Maps data to key/value pairs
Reduces intermediate results to final output
Largely supplanted by Spark these days

Yet Another Resource Negotiator
Manages cluster resources for multiple data processing frameworks

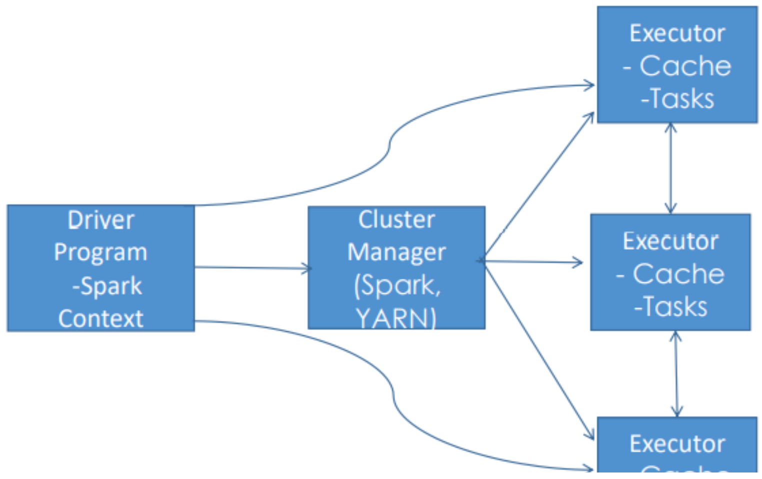
Hadoop Distributed File System
Distributes data blocks across cluster in a redundant manner
Ephemeral in EMR; data lost on termination

Apache Spark

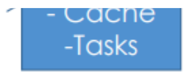


- Distributed processing framework for big data
- In-memory caching, optimized query executio
- Supports Java, Scala, Python, and R
- Supports code reuse across
  - Batch processing
  - Interactive Queries
    - Spark SQL
  - Real-time Analytics
  - Machine Learning
    - MLLib
  - Graph Processing
- Spark Streaming
  - Integrated with Kinesis, Kafka, on EMR
- Spark is NOT meant for OLTP

How Spark Works



- Spark apps are run as independent processes on a cluster
- The SparkContext (driver program) coordinates them
- SparkContext works through a Cluster Manager
- Executors run computations and store data
- SparkContext sends application code and tasks to executors


- Cache
-Tasks