**AIM**

Recognize use cases for Pig, which include:

• ETL data pipelines

• Researching raw data

• Iterative data processing

**OVERVIEW**

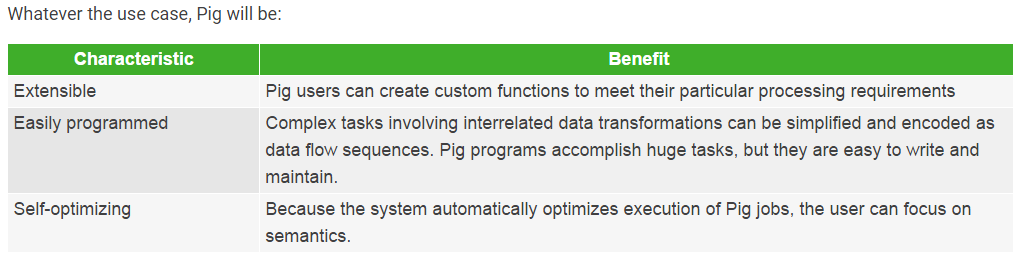
A scripting platform for processing and analyzing large data sets

With YARN as the architectural center of ApacheTM Hadoop, multiple data access engines such as Apache Pig interact with data stored in the cluster. Apache Pig allows Apache Hadoop users to write complex MapReduce transformations using a simple scripting language called Pig Latin. Pig translates the Pig Latin script into MapReduce so that it can be executed within YARN for access to a single dataset stored in the Hadoop Distributed File System (HDFS).

**WHAT PIG DOES**

Pig was designed for performing a long series of data operations, making it ideal for three categories of Big Data jobs:

* **Extract-transform-load (ETL)** data pipelines,
* **Research** on raw data, and
* **Iterative data processing**



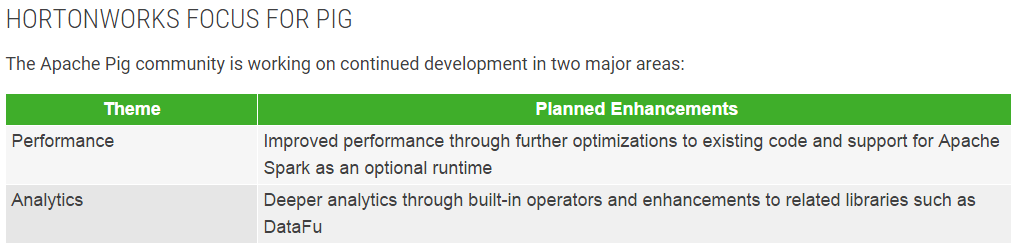
**HOW PIG WORKS**

Pig runs on Apache Hadoop YARN and makes use of MapReduce and the Hadoop Distributed File System (HDFS). The language for the platform is called Pig Latin, which abstracts from the Java MapReduce idiom into a form similar to SQL. While SQL is designed to query the data, Pig Latin allows you to write a data flow that describes how your data will be transformed (such as aggregate, join and sort).

Since Pig Latin scripts can be graphs (instead of requiring a single output) it is possible to build complex data flows involving multiple inputs, transforms, and outputs. Users can extend Pig Latin by writing their own functions, using Java, Python, Ruby, or other scripting languages. Pig Latin is sometimes extended using UDFs (User Defined Functions), which the user can write in any of those languages and then call directly from the Pig Latin.

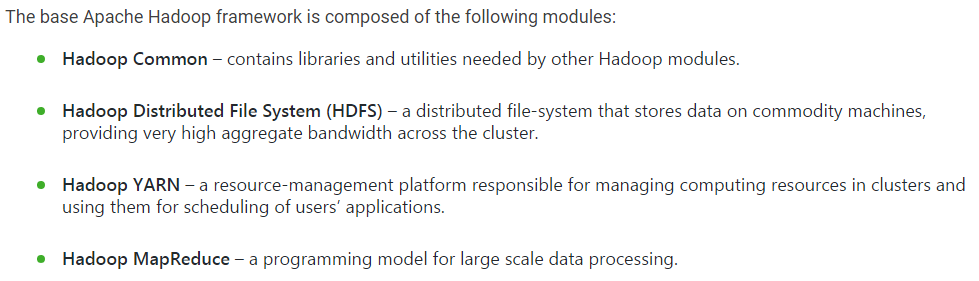
The user can run Pig in two modes, using either the “pig” command or the “java” command:

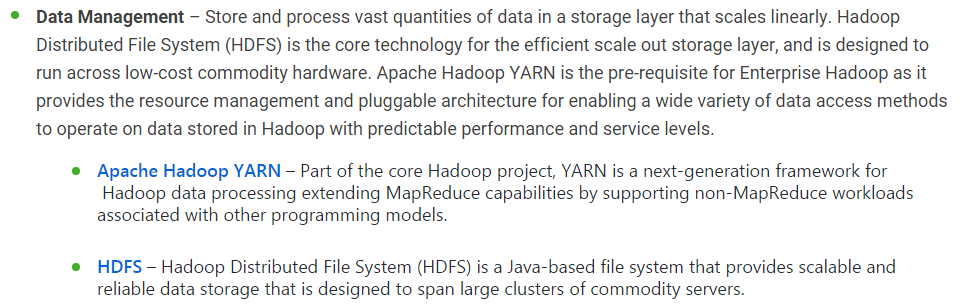
* **MapReduce Mode.** This is the default mode, which requires access to a Hadoop cluster.
* **Local Mode.** With access to a single machine, all files are installed and run using a local host and file system.



**HDP 2.4 – Pig 0.15.0**

* Pig on Tez stablization
* Improved Tez auto-parallelism
* Ability to invoke Hive UDFs from Pig





* [**Apache Hive**](https://hortonworks.com/hadoop/hive) – Built on the MapReduce framework, Hive is a data warehouse that enables easy data summarization and ad-hoc queries via an SQL-like interface for large datasets stored in HDFS.
* [**Apache Pig**](https://hortonworks.com/hadoop/pig) – A platform for processing and analyzing large data sets. Pig consists of a high-level language (Pig Latin) for expressing data analysis programs paired with the MapReduce framework for processing these programs.
* [**MapReduce**](https://hortonworks.com/hadoop/mapreduce/) – MapReduce is a framework for writing applications that process large amounts of structured and unstructured data in parallel across a cluster of thousands of machines, in a reliable and fault-tolerant manner.
* [**Apache Spark**](https://hortonworks.com/hadoop/spark) – Spark is ideal for in-memory data processing. It allows data scientists to implement fast, iterative algorithms for advanced analytics such as clustering and classification of datasets.
* [**Apache Storm**](https://hortonworks.com/hadoop/storm) – Storm is a distributed real-time computation system for processing fast, large streams of data adding reliable real-time data processing capabilities to Apache Hadoop 2.x
* [**Apache HBase**](https://hortonworks.com/hadoop/hbase) – A column-oriented NoSQL data storage system that provides random real-time read/write access to big data for user applications.
* [**Apache Tez**](https://hortonworks.com/hadoop/tez) – Tez generalizes the MapReduce paradigm to a more powerful framework for executing a complex DAG (directed acyclic graph) of tasks for near real-time big data processing.
* [**Apache Kafka**](https://hortonworks.com/hadoop/kafka) – Kafka is a fast and scalable publish-subscribe messaging system that is often used in place of traditional message brokers because of its higher throughput, replication, and fault tolerance.
* [**Apache HCatalog**](https://hortonworks.com/hadoop/hcatalog) – A table and metadata management service that provides a centralized way for data processing systems to understand the structure and location of the data stored within Apache Hadoop.
* [**Apache Slider**](https://hortonworks.com/hadoop/slider) – A framework for deployment of long-running data access applications in Hadoop. Slider leverages YARN’s resource management capabilities to deploy those applications, to manage their lifecycles and scale them up or down.
* [**Apache Solr**](https://hortonworks.com/hadoop/solr) – Solr is the open source platform for searches of data stored in Hadoop. Solr enables powerful full-text search and near real-time indexing on many of the world’s largest Internet sites.
* [**Apache Mahout**](https://hortonworks.com/hadoop/mahout) – Mahout provides scalable machine learning algorithms for Hadoop which aids with data science for clustering, classification and batch based collaborative filtering.
* [**Apache Accumulo**](https://hortonworks.com/hadoop/accumulo) – Accumulo is a high performance data storage and retrieval system with cell-level access control. It is a scalable implementation of Google’s Big Table design that works on top of Apache Hadoop and Apache ZooKeeper.

**Operations** – Provision, manage, monitor and operate Hadoop clusters at scale.

* [**Apache Ambari**](https://hortonworks.com/hadoop/ambari) – An open source installation lifecycle management, administration and monitoring system for Apache Hadoop clusters.
* [**Apache Oozie**](https://hortonworks.com/hadoop/oozie) – Oozie Java Web application used to schedule Apache Hadoop jobs. Oozie combines multiple jobs sequentially into one logical unit of work.
* [**Apache ZooKeeper**](https://hortonworks.com/hadoop/zookeeper) – A highly available system for coordinating distributed processes. Distributed applications use ZooKeeper to store and mediate updates to important configuration information.

An HDFS cluster is comprised of a **NameNode**, which manages the cluster metadata, and DataNodes that store the data. Files and directories are represented on the NameNode by inodes. Inodes record attributes like permissions, modification and access times, or namespace and disk space quotas.

The file content is split into large blocks (typically 128 megabytes), and each block of the file is independently replicated at multiple DataNodes. The blocks are stored on the local file system on the DataNodes.

The Namenode actively monitors the number of replicas of a block. When a replica of a block is lost due to a DataNode failure or disk failure, the NameNode creates another replica of the block. The NameNode maintains the namespace tree and the mapping of blocks to DataNodes, holding the entire namespace image in RAM.

The NameNode does not directly send requests to DataNodes. It sends instructions to the DataNodes by replying to heartbeats sent by those DataNodes. The instructions include commands to:

* replicate blocks to other nodes,
* remove local block replicas,
* re-register and send an immediate block report, or
* shut down the node.