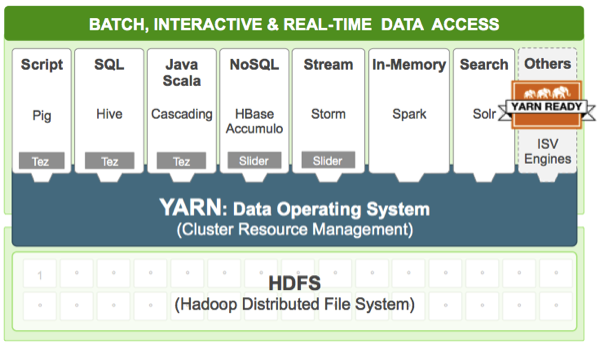
# What YARN does

<http://hortonworks.com/apache/yarn/>

Part of the core Hadoop project, YARN is the architectural center of Hadoop that allows multiple data processing engines such as interactive SQL, real-time streaming, data science and batch processing to handle data stored in a single platform, unlocking an entirely new approach to analytics.

YARN is the prerequisite for Enterprise Hadoop, providing resource management and a central platform to deliver consistent operations, security, and data governance tools across Hadoop clusters.



As its architectural center, YARN enhances a Hadoop compute cluster in the following ways:

| **Feature** | **Description** |
| --- | --- |
| Multi-tenancy | YARN allows multiple access engines (either open-source or proprietary) to use Hadoop as the common standard for batch, interactive and real-time engines that can simultaneously access the same data set. |
| Cluster utilization | YARN’s dynamic allocation of cluster resources improves utilization over more static MapReduce rules used in early versions of Hadoop |
| Scalability | Data center processing power continues to rapidly expand. YARN’s ResourceManager focuses exclusively on scheduling and keeps pace as clusters expand to thousands of nodes managing petabytes of data. |
| Compatibility | Existing MapReduce applications developed for Hadoop 1 can run YARN without any disruption to existing processes that already work |

# How YARN Works

YARN’s original purpose was to split up the two major responsibilities of the JobTracker/TaskTracker into separate entities:

* a global ResourceManager
* a per-application ApplicationMaster
* a per-node slave NodeManager
* a per-application Container running on a NodeManager

The ResourceManager and the NodeManager formed the new generic system for managing applications in a distributed manner. The ResourceManager is the ultimate authority that arbitrates resources among all applications in the system. The ApplicationMaster is a framework-specific entity that negotiates resources from the ResourceManager and works with the NodeManager(s) to execute and monitor the component tasks.

The ResourceManager has a scheduler, which is responsible for allocating resources to the various applications running in the cluster, according to constraints such as queue capacities and user limits. The scheduler schedules based on the resource requirements of each application.

Each ApplicationMaster has responsibility for negotiating appropriate resource containers from the scheduler, tracking their status, and monitoring their progress. From the system perspective, the ApplicationMaster runs as a normal container.

The NodeManager is the per-machine slave, which is responsible for launching the applications’ containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager.