Les différents cluster Managers

Les différents types de gestionnaire de cluster

- Le système prend actuellement en charge plusieurs gestionnaires de cluster:
  - **Standalone -** un gestionnaire de cluster simple inclus avec Spark qui facilite la configuration d'un cluster
  - **Apache Mesos -** Un gestionnaire de cluster général qui peut également exécuter Hadoop mapReduce et des applications de service
  - **Hadoop Yarn -** le gestionnaire de ressources dans Hadoop 2 et 3
  - **Kubernetes -** un système opensource pour automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées.

Standalone

Le gestionnaire de cluster Spark Standalone est un simple gestionnaire de cluster intégré à Spark. Il facilite la configuration d'un cluster que Spark gère lui-même et peut s'exécuter sous

- Linux ,
- Windows
- ou Mac OSX.

C'est souvent le moyen le plus simple d'exécuter une application Spark dans un environnement en cluster.

Standalone

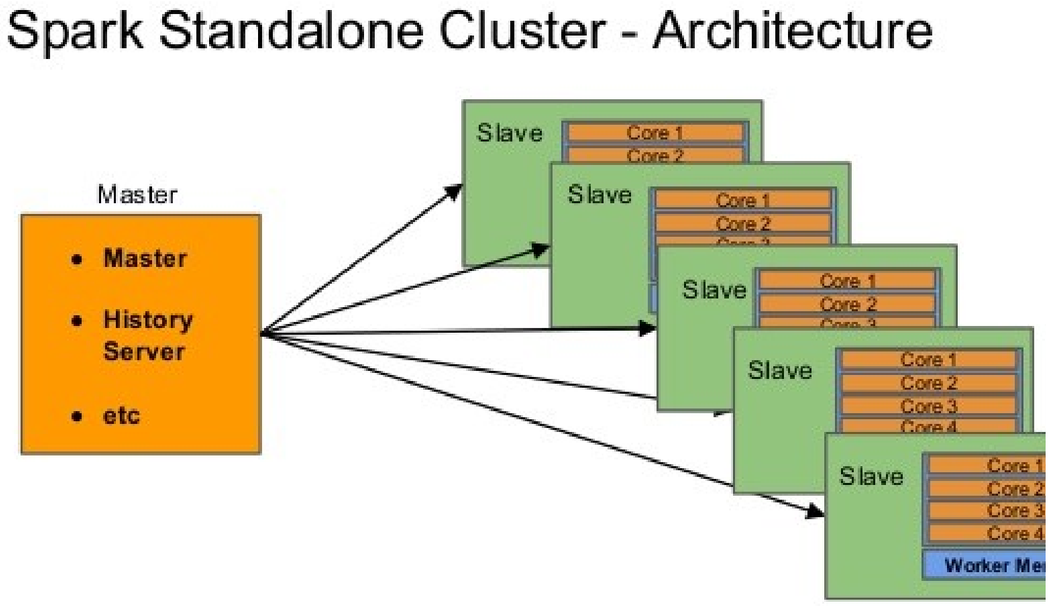
Comment fonctionne le cluster Spark standalone ?

Il a des *maîtres* et un nombre de *travailleurs* avec une quantité configurée de mémoire et de cœurs de processeur.

En mode cluster Spark standalone , Spark alloue des ressources en fonction des coeurs et de la mémoire.

Dans le gestionnaire de cluster Spark Standalone, **Zookeeper** récupère le maître à l'aide du maître de *secours.* En utilisant le système de fichiers, nous pouvons réaliser la récupération manuelle du maître.

Dans ce gestionnaire de cluster, nous avons une interface utilisateur Web pour afficher les statistiques de cluster et de travail.



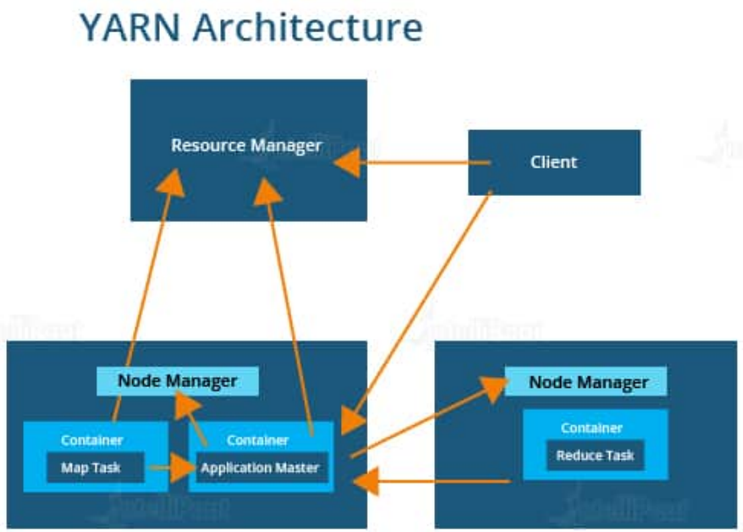
Yarn

YARN est devenu le sous-projet de Hadoop en 2012.

YARN divise les fonctionnalités du gestionnaire de ressources et de la planification des tâches en différents démons.

Le plan est d'obtenir un gestionnaire de ressources global (RM) et un maître d'application (AM) par application. Une application est soit un DAG de graphe, soit une tâche individuelle.

Yarn



Le framework de calcul de données YARN est une combinaison du *ResourceManager* , du *NodeManager* .

Le Yarn Resource Manager gère les ressources parmi toutes les applications du système. Le gestionnaire de ressources a

- un planificateur
- et un gestionnaire d'applications.

Le planificateur alloue des ressources aux différentes applications en cours d'exécution. C'est un pur planificateur, effectue une surveillance ou un suivi de l'état de l'application.

Le gestionnaire d'applications gère les applications sur tous les nœuds.

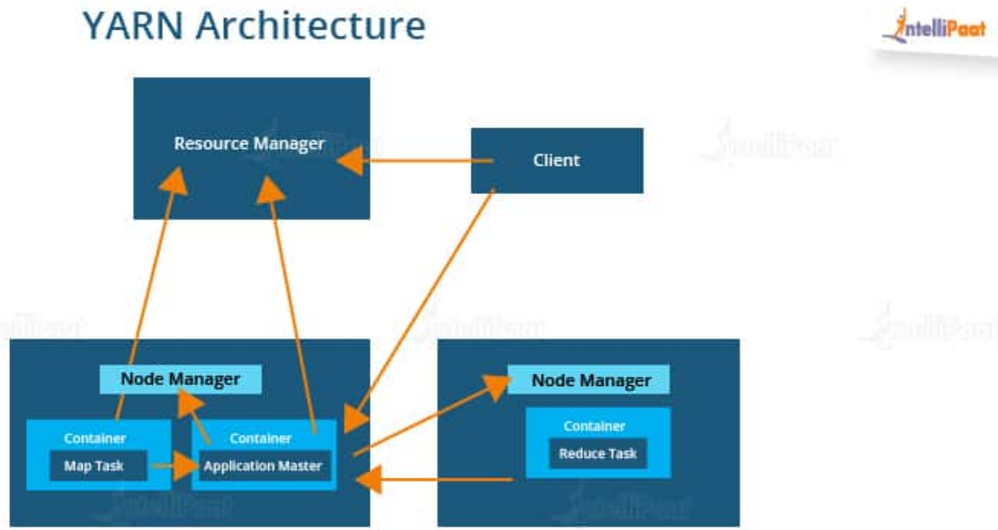
Yarn

Yarn Node Manager contient

- Application Master
- et conteneur.

Un conteneur est un lieu où se produit une unité de travail. Chaque tâche de MapReduce s'exécute dans un conteneur. Le maître d'application par application est une bibliothèque spécifique au framework. Il vise à négocier les ressources auprès du Resource Manager. Il continue avec le ou les gestionnaires de nœuds pour exécuter et surveiller les tâches.

L'application où le travail nécessite un ou plusieurs conteneurs. Node Manager gère les conteneurs de surveillance, l'utilisation des ressources (CPU, mémoire, disque et réseau). Il en fait rapport au gestionnaire de ressources.



Mesos

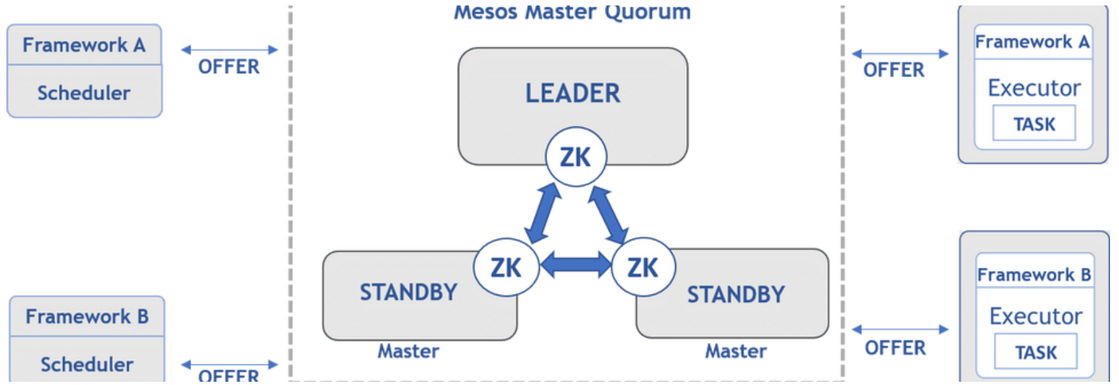
Apache Mesos gère la charge de travail à partir de nombreuses sources en utilisant le partage et l'isolement dynamiques des ressources. Il aide au déploiement et à la gestion des applications dans des environnements de cluster à grande échelle.

Apache Mesos se compose de trois composants :

- **Mesos Master** : Mesos Master offre une tolérance aux pannes (la capacité de fonctionner et de récupérer la perte en cas de panne). Un cluster contient de nombreux maîtres Mesos.

- **Mesos Slave** : Mesos Slave est une instance qui offre des ressources au cluster. Mesos Slave affecte des ressources uniquement lorsqu'un Mesos Master affecte une tâche.

- **Frameworks Mesos** : Les Frameworks Mesos permettent aux applications de demander des ressources au cluster afin que l'application puisse effectuer les tâches.



Modes d'exécution

Modes d'exécution

Un modèle d'exécution aide à déterminer où se trouvent physiquement les ressources mentionnées précédemment lorsque l'application est exécutée. Vous avez le choix entre trois modes d'exécution :

- Mode Cluster
- Mode Client
- Mode local

Mode Cluster

Mode cluster

Le mode cluster est le moyen le plus courant d'exécuter des applications Spark au cours duquel un script Python, JAR ou R précompilé est soumis à un gestionnaire de cluster par un utilisateur.

Le processus pilote est ensuite lancé sur un nœud de travail à l'intérieur du cluster par le gestionnaire de cluster, en plus des processus exécuteurs.

Cela implique que le gestionnaire de cluster est en charge de la maintenance de tous les processus liés à l'application Spark.

Mode Client

Mode client

Le mode client est presque identique au mode cluster, sauf que le pilote Spark reste sur la machine cliente qui a soumis l'application.

Cela signifie que la machine cliente gère le processus du pilote Spark et que le gestionnaire de cluster gère ceux de l'exécuteur.

Ces machines sont communément appelées machines passerelles ou nœuds périphériques

Mode local

Mode local

En mode local, l'intégralité de l'application Spark est exécutée sur une seule machine. Il observe le parallélisme à travers les threads sur cette seule machine.

Il s'agit d'un moyen courant de tester des applications ou d'expérimenter le développement local.

Cependant, il n'est pas recommandé pour l'exécution d'applications de production.

Mode local

Mode local

En mode local, l'intégralité de l'application Spark est exécutée sur une seule machine. Il observe le parallélisme à travers les threads sur cette seule machine.

Il s'agit d'un moyen courant de tester des applications ou d'expérimenter le développement local.

Cependant, il n'est pas recommandé pour l'exécution d'applications de production.