**底层框架介绍**

**k8s**

[k8s介绍](https://kubernetes.io/zh-cn/docs/concepts/overview/)

Kubernetes 这个名字源于希腊语，意为“舵手”或“飞行员”。k8s 这个缩写是因为 k 和 s 之间有八个字符的关系。Kubernetes 是一个可移植、可扩展的开源平台，用于管理容器化的工作负载和服务，可促进声明式配置和自动化。

容器是打包和运行应用程序的好方式。容器类似于 VM，但是更宽松的隔离特性，使容器之间可以共享操作系统（OS）。 因此，容器比起 VM 被认为是更轻量级的。且与 VM 类似，每个容器都具有自己的文件系统、CPU、内存、进程空间等。 由于它们与基础架构分离，因此可以跨云和 OS 发行版本进行移植。

在生产环境中， 你需要管理运行着应用程序的容器，并确保服务不会下线。 例如，如果一个容器发生故障，则你需要启动另一个容器。 这就是 Kubernetes 帮你做的事情！ Kubernetes 为你提供了一个可弹性运行分布式系统的框架。 Kubernetes 会满足你的扩展要求、故障转移你的应用、提供部署模式等。

**Spark on k8s**

[Spark on k8s介绍1](https://blog.csdn.net/zfw_666666/article/details/127512994)

[Spark on k8s介绍2](https://zhuanlan.zhihu.com/p/66675447)

通过 spark-submit 直接向 K8S 的 API Server 提交，申请到资源后启动 Pod 做为 Driver 和 Executor 执行 Job

工作原理如下：

* Spark创建一个Spark driver 在Kubernetes pod 运行。
* driver 程序创建也在Kubernetes pods中运行的executor，并连接到它们，然后执行应用程序代码。
* 当应用程序完成时，执行程序pod将终止并被清理，但驱动程序pod会保存日志并在Kubernetes API中保持“已完成”状态，直到最终进行垃圾收集或手动清理。

**submit Python to Spark**

以提交fpgrowth.py代码为例

* 在文件夹/nfsdata1下编写fpgrowth.py代码
* 输入文件在data
* 输出文件在output
* 日志文件在log

|  |
| --- |
| Python from pyspark.ml.fpm import FPGrowth from pyspark import SparkContext from pyspark.sql import SparkSession *# df = spark.createDataFrame([ # (0, [1, 2, 5]), # (1, [1, 2, 3, 5]), # (2, [1, 2]) # ], ["id", "items"])* sc = SparkContext(appName="FPGrowth") spark = SparkSession(sc) data = sc.textFile("/mydata/data/data2") transactions = data.map(lambda line: (line.strip().split(' '),)) df = transactions.toDF(["items",]) fpGrowth = FPGrowth(itemsCol="items", minSupport=4e-6, minConfidence=0.4) model = fpGrowth.fit(df)  *# Display frequent itemsets. # model.freqItemsets.show()*  *# Display generated association rules.* answer=model.associationRules from pyspark.sql.types import \* from pyspark.sql.functions import udf def **str\_to\_arr**(my\_list):  my\_list.sort()  return '+'.join([str(elem) for elem in my\_list])  str\_to\_arr\_udf = udf(str\_to\_arr,StringType()) answer = answer.withColumn("antecedent",str\_to\_arr\_udf(answer["antecedent"])) answer = answer.withColumn("consequent",str\_to\_arr\_udf(answer["consequent"])) answer.write.option("header",True).csv("/mydata/output/data\_support\_4e-6\_confidence\_0.4")  *# transform examines the input items against all the association rules and summarize the # consequents as prediction #model.transform(df).show()* spark.stop() |

* 获取k8s所有容器

|  |
| --- |
| Bash kubectl get pods --all-namespaces |

* 进入某个spark-master
* 根据get pods输出，找到名称和命名空间
* 名称：spark-master-controller-bbxvq
* 命名空间：spark-cluster

|  |
| --- |
| Bash kubectl exec -it spark-master-controller-bbxvq -n spark-cluster -- /bin/bash |

* 运行fpgrowth.py
* 通过kubectl cluster-info获取：master k8s://https://10.176.62.221:6443
* 部署模式为集群：deploy-mode cluster
* 设定每个job的executor运行个数为5：spark.executor.instances=5
* 根据spark配置文件设置用户：spark.kubernetes.authenticate.driver.serviceAccountName=spark
* 根据get pods获取namespace：spark.kubernetes.namespace=spark-cluster
* 自定义driver的名字为spark-wctest-1：spark.kubernetes.driver.pod.name=spark-wctest-1
* 设置镜像：spark.kubernetes.container.image=awayee/spark-py:add\_numpy
* 设置driver的pod的文件夹（共享映射到pvc所配置的/nfsdata1）：spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.path=/mydata
* 设置driver的文件夹为可读可写：spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.readOnly=false
* 配置driver的文件中的pvc名称：spark.kubernetes.driver.volumes.persistentVolumeClaim.data.options.claimName=mypvc2
* 设置executor的pod的文件夹（共享映射到pvc所配置的/nfsdata1）：spark.kubernetes.executor.volumes.persistentVolumeClaim.data.mount.path=/mydata
* 设置executor的文件夹为可读可写：spark.kubernetes.executor.volumes.persistentVolumeClaim.data.mount.readOnly=false
* 配置executor的文件中的pvc名称：spark.kubernetes.executor.volumes.persistentVolumeClaim.data.options.claimName=mypvc2
* 指定job的名称：name spark\_wc\_1
* 指定程序（/mydata已映射到/nfsdata1）：local:///mydata/fpgrowth.py

|  |
| --- |
| Bash /opt/spark/bin/spark-submit \       --master k8s://https://10.176.62.221:6443 \       --deploy-mode cluster \       --conf spark.executor.instances=5 \       --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \       --conf spark.kubernetes.namespace=spark-cluster \       --conf spark.kubernetes.driver.pod.name=spark-wctest-1 \       --conf spark.kubernetes.container.image=awayee/spark-py:add\_numpy\       --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.path=/mydata \       --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.readOnly=false \       --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.options.claimName=mypvc2 \       --conf spark.kubernetes.executor.volumes.persistentVolumeClaim.data.mount.path=/mydata \       --conf spark.kubernetes.executor.volumes.persistentVolumeClaim.data.mount.readOnly=false \       --conf spark.kubernetes.executor.volumes.persistentVolumeClaim.data.options.claimName=mypvc2 \       --conf spark.executor.heartbeatInterval=100s\       --name spark\_wc\_1 \ local:///mydata/fpgrowth.py |

* 退出pod

|  |
| --- |
| Bash exit |

* 查看运行日志，输出到当前目录的out.txt下
* 通过运行输出的日志末尾，找到driver名称
* driver名称：spark-wctest-1

|  |
| --- |
| Bash kubectl logs -f spark-wctest-1 -n spark-cluster > log/log.txt |

* 删除用完的driver
* driver状态：Completed
* 也可以删除其他状态

|  |
| --- |
| Bash kubectl get pods -n spark-cluster | grep Completed | awk '{print$1}' | xargs kubectl delete -n spark-cluster pods |