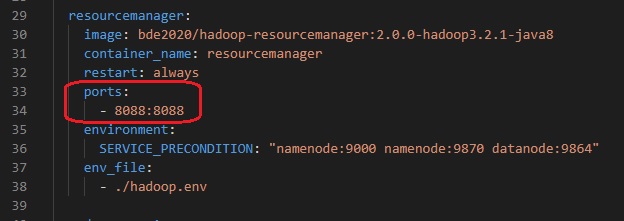
Всем привет, пришло время 3го ДЗ:

1. Ознакомится с типовыми задачами MapReduce: https://docs.microsoft.com/ru-ru/azure/hdinsight/hadoop/apache-hadoop-run-samples-linux.

Прописать в docker-comose порт (8088:8088) для resourcemanager (пример приложил в материалы).

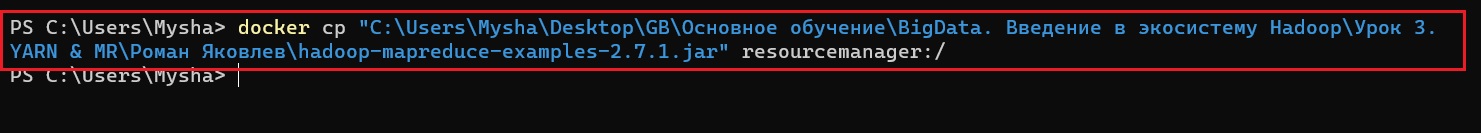
прописал вот:



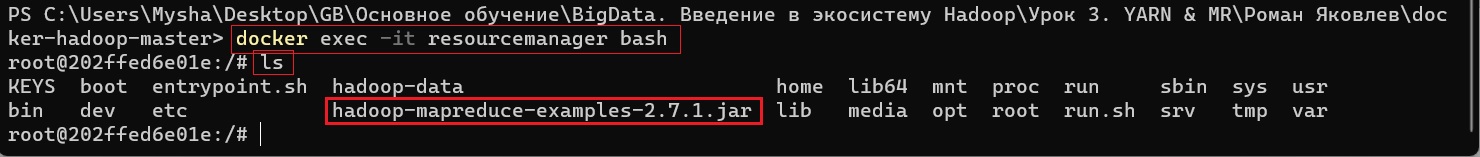
Пересобрать контейнеры и запустить задачу из примеров, например, вычисление pi методом Монте-Карло

\* Скопировать /hadoop-mapreduce-examples-2.7.1.jar в контейнер и запустить: yarn jar /hadoop-mapreduce-examples-2.7.1.jar pi 1 10

Передаём в корень контейнера .jar-файл

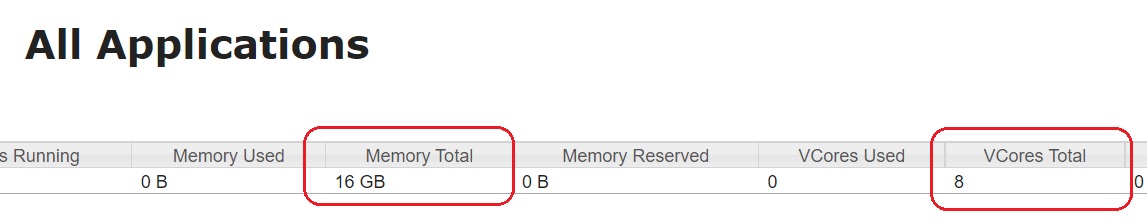


Вот он лежит:



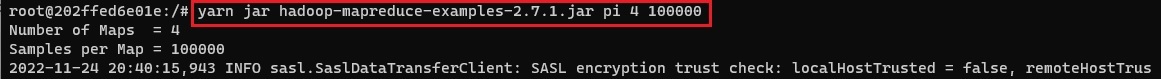
зайти на ResourceManager http://localhost:8088/ и найти свою задачу и сказать сколько ресурсов она занимает при выполнении, сколько времени она выполнялась и на каких нодах (лучше просто скрины, где эта информация выделена) + просто посёрфить по веб-модре ярна и посмотреть, что там есть

а вот наши доступные ресурсы на старте:

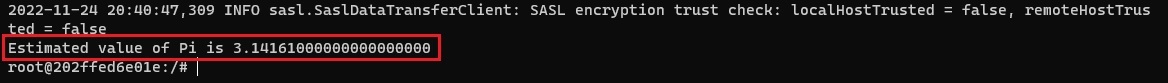


Запускаем подсчёт pi, поставим 4 карты на 100000 точек каждая

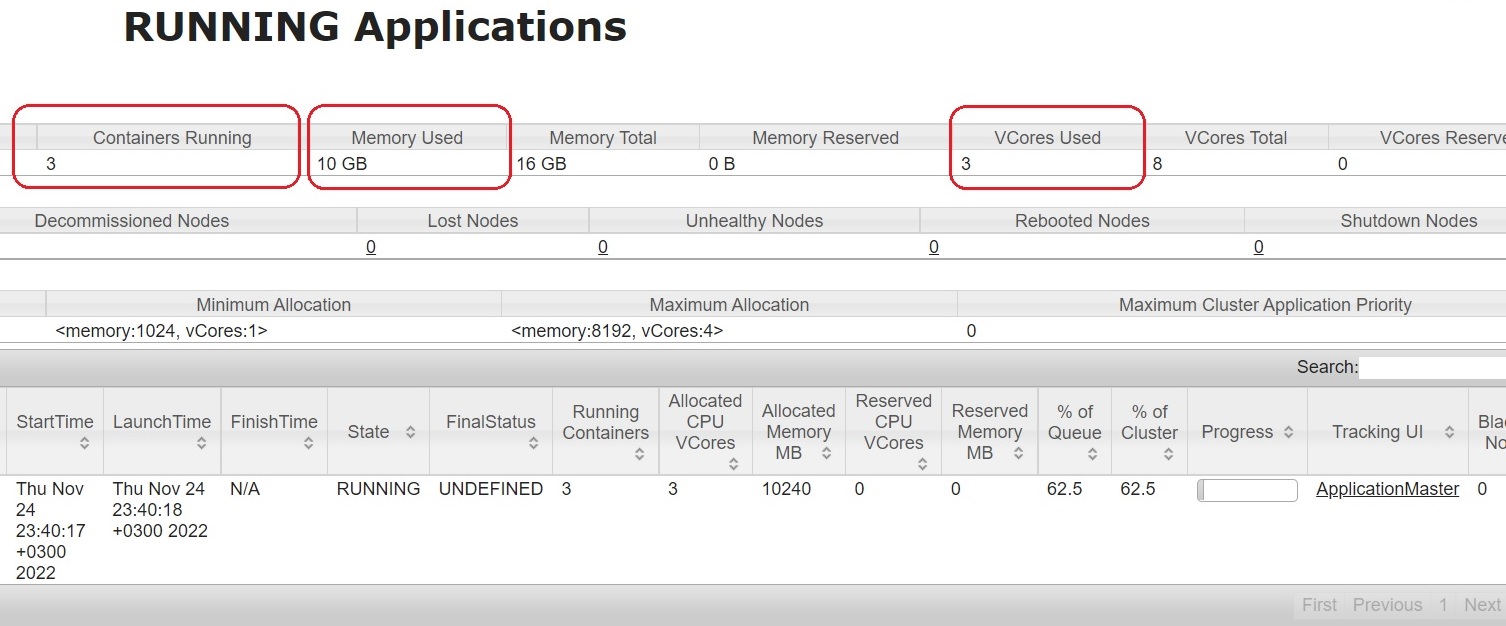
1) запустились



2) получили результат



3) а вот что в это время происходило в localhost – 3 контейнера работало, 10GB памяти было занято и 3 ядра



4) время – 27 секунд, 257381 МБ-секунд и 69 ядер-секунд

Изображение выглядит как стол

Автоматически созданное описание

Задача 2 не обязательна, вводные для неё дам на сл лекции.

\*2. Реализовать алгоритм на MapReduce на python или bash который сделает следующее: данные из task3.txt при проходе через маппер, должны отчищаться от букв, а на стадии редьюс должны выводиться батчами по 1-5 штук (рандомно) через запятую. Прислать сюда код и вывод. Можно так же просто запустить через bash. Пример вывода:

[student41\_35@bigdataanalytics2-worker-shdpt-v31-1-0 hm3]$ head -100 task | ./mapper.py | ./reducer.py

311700742

5711309,074885209

533403406118,282041732,514126202079

09055027727,05059025,23688850331,005246958,177406

963409798683,108092297,748536185,144964208,740692150

11443366871,8737674,843390903229,137952503

97015724537,8845439897,0807534495,5158867661

16131379

4437031881,44425467

6102238076

1424362,492928437,250286840435,355869223055,30954768

Задачи со \* предназначены для продвинутых учеников, которым мало сделать обычное ДЗ.

1) на локалке создадим папку и положим в неё файл task\_3.txt, работать будем из этой папки

mkdir dz\_3

cd dz\_3

2) отправим файл на HDFS в такую же папку, чтоб без папки не валялся

hdfs dfs -put task\_3.txt dz\_3/

3) создадим два .py-файла и дадим им права на исполнение

chmod u+x mapper.py

chmod u+x reducer.py

4) в mapper.ry от хешей оставляем только цифры

5) в reducer.py получаемые цифры собираем в рандомные списки длиной от 1-го до 5-ти элементов и сразу же выводим их на печать, приджойнив значения через запятую

В этом же скрипте оставил два счётчика: count и batch\_count. Count считает количество входящих хешей, а batch\_count сумму элементов в формируемых списках. Числа в итоге получаются одинаковыми по 50000, а значит всё отработало хорошо и ничего не потерялось.

6) а запускать MapReduce будем так:

hdfs dfs -cat dz\_3/task\_3.txt | ./mapper.py | ./reducer.py

и результат работы на скрине

