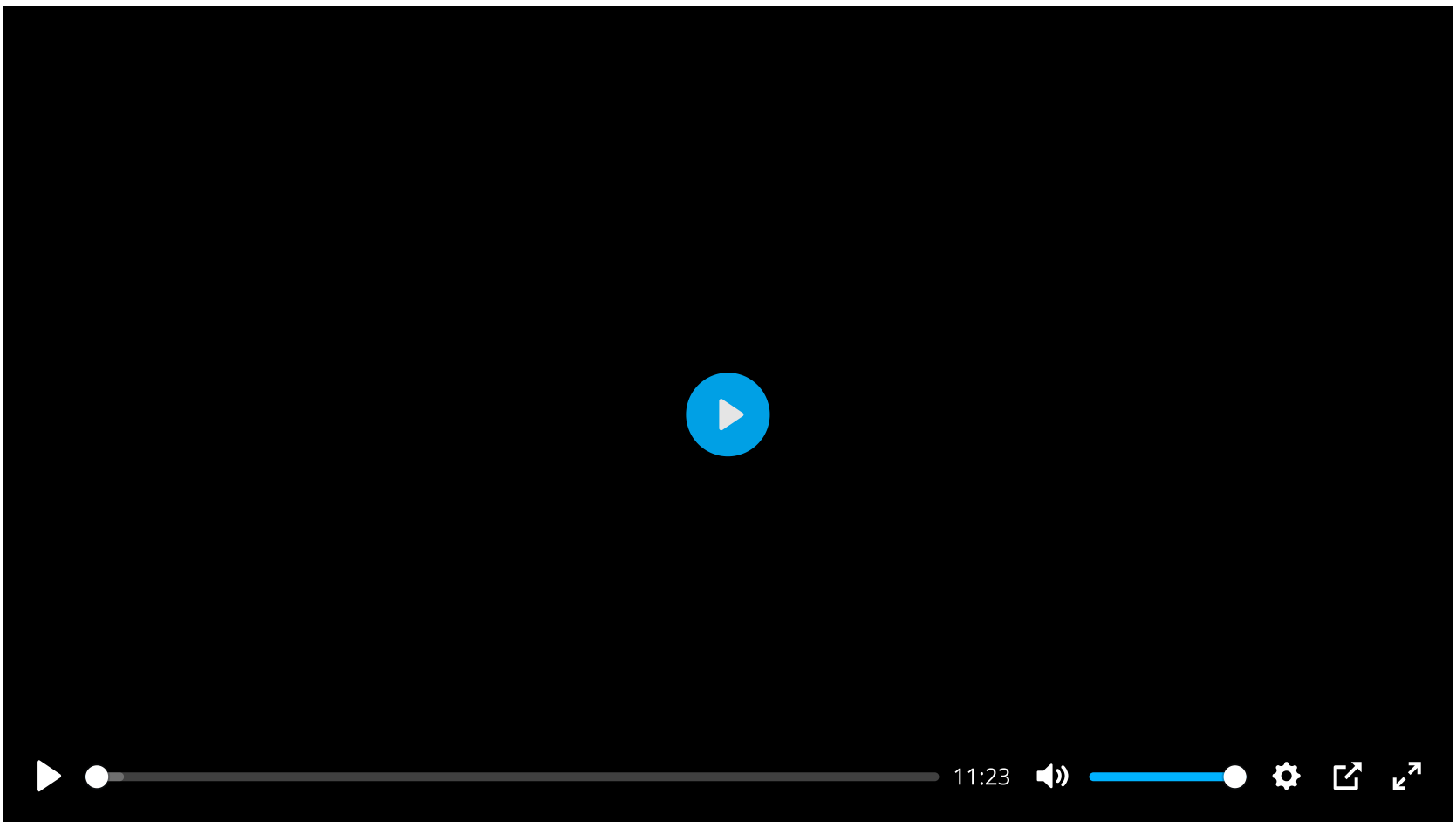
3.3 Технология Map-Reduce



Текстовая расшифровка видео:

КАК РАБОТАЮТ JOIN'Ы

План:

- Задача: средняя оценка по предмету;
- MapJoin.

Задача: средняя оценка по предмету

Задача:

Есть табель и результат опроса студентов: «Какой предмет самый любимый». Как посчитать для каждого предмета среднюю оценку среди любителей этого предмета?

Нам может помочь **ReduceJoin**.

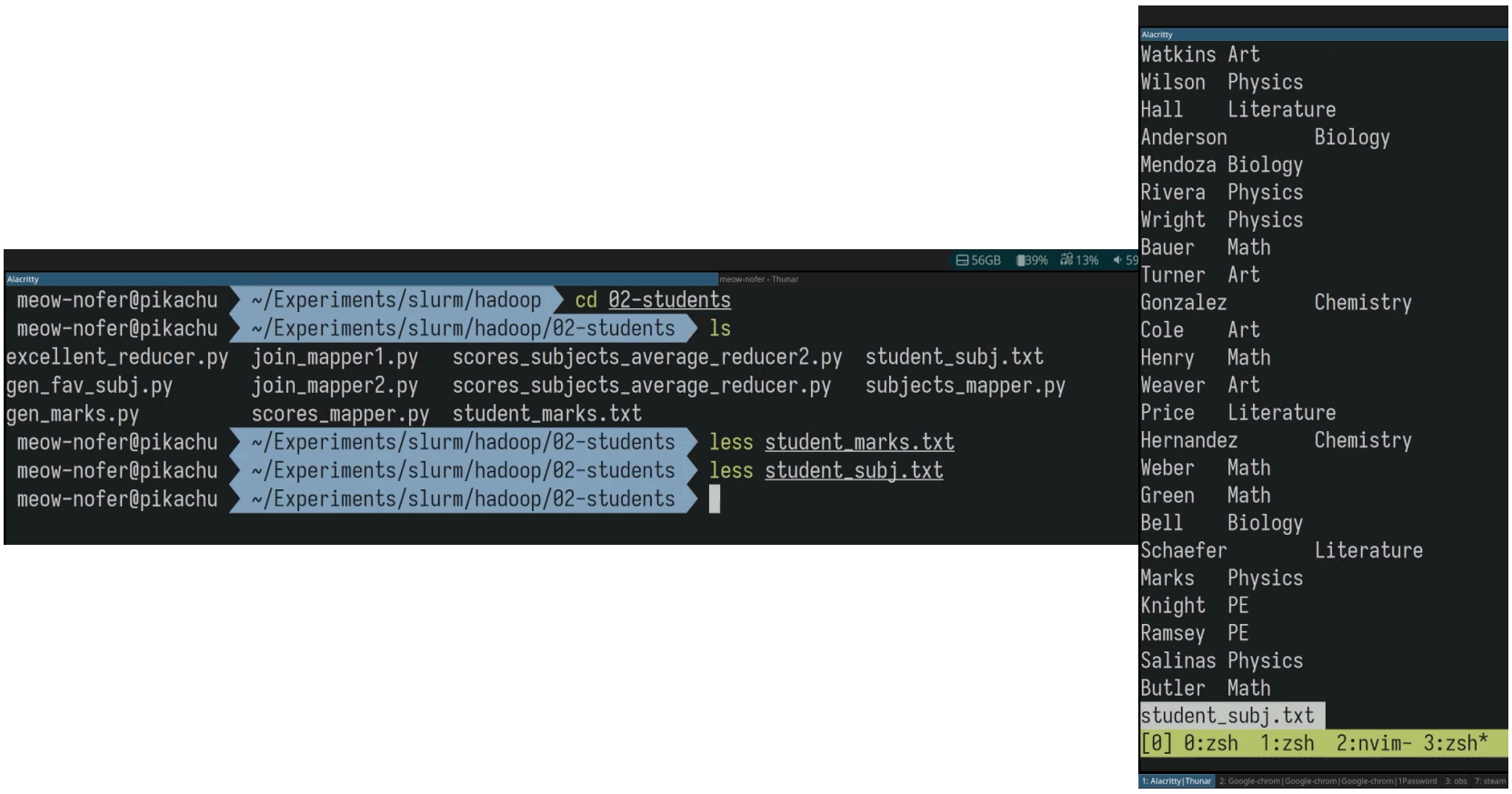
Делаем для каждого датасета составной ключ с типом значений. Считаем для каждого предмета среднюю оценку для каждого студента. Считаем среднюю оценку по всем студентам.

Подробнее:

На входе у нас есть два датасета. Если мы можем определить по содержимому файлу, с чем работаем, то мы можем решить все в маппере либо сделать две Map Only-Job, каждая из которых будет читать свои файлы.

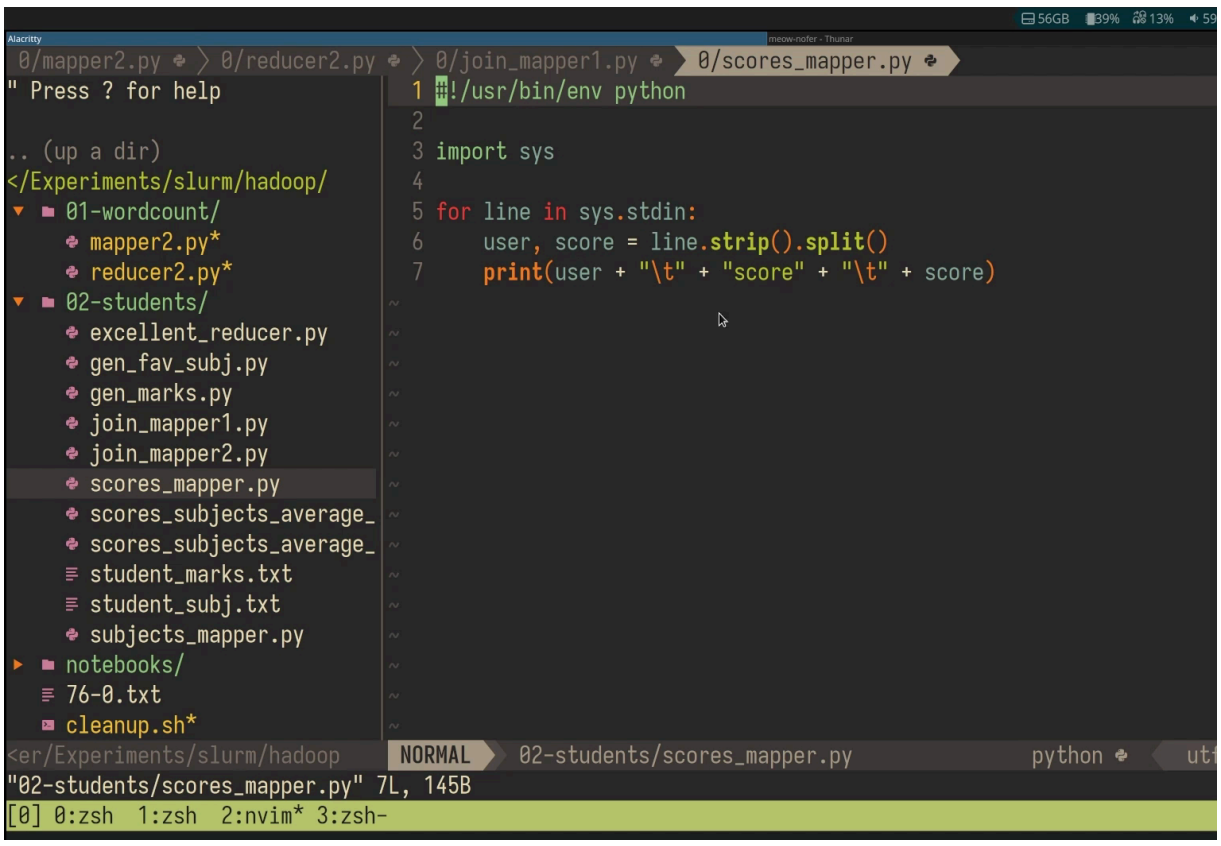
На входе у нас есть два датасета. Если по файлу нам известно содержание, то мы можем работать в маппере. Другой вариант – сделать две Map Only-Job, задача которых читать свои файлы.

Рассмотрим на примере:



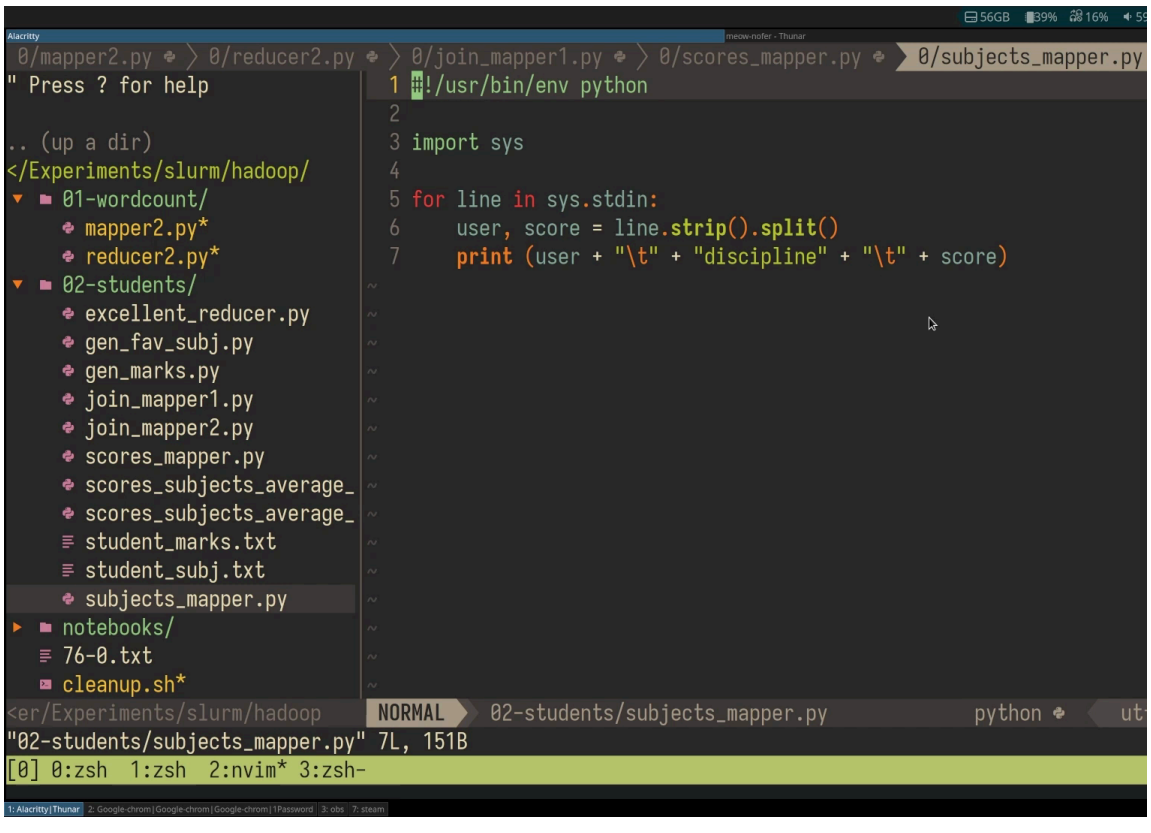
У нас есть файлы с оценками и любимыми предметами. Наша задача – их смапить.

Когда мы читаем оценки студента в маппере, мы можем писать следующее:



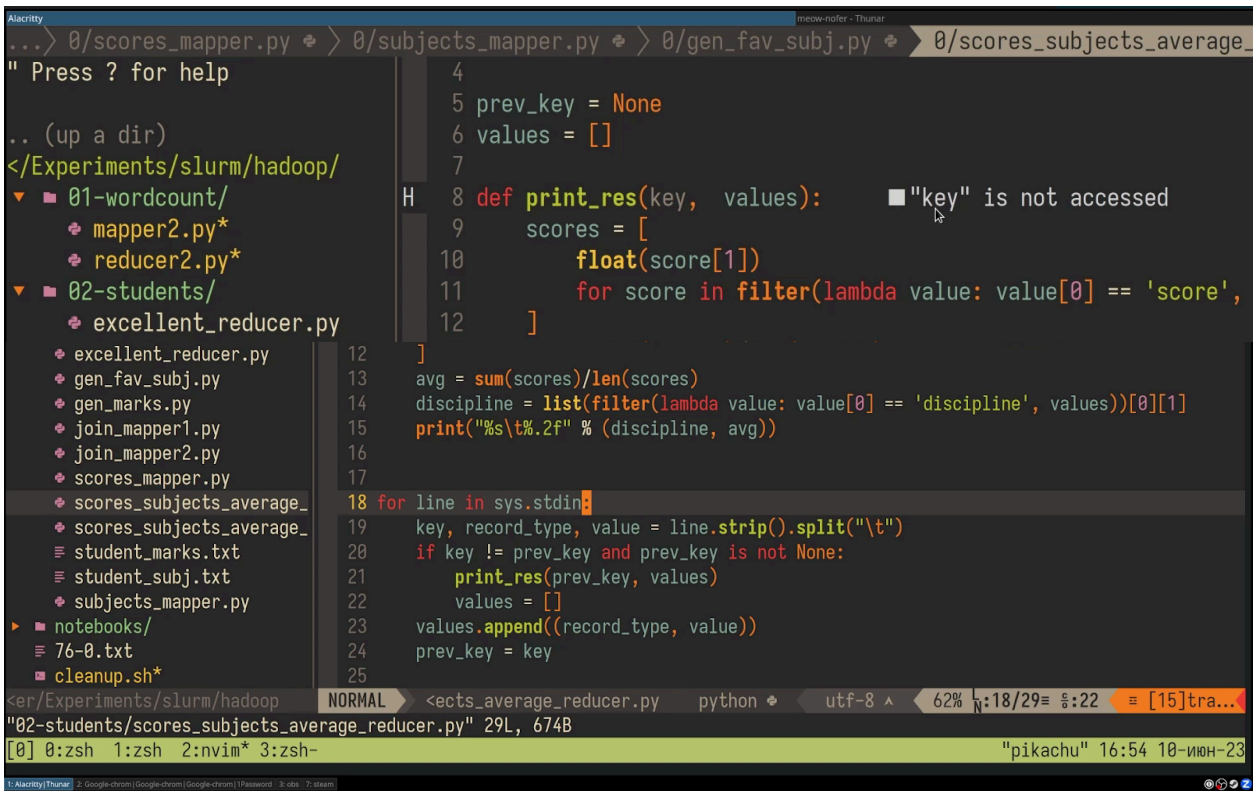
«для такого-то студента»; добавляем колонку с типом данных (указать, что это «score» (оценка)); через tab добавляем саму оценку.

Для любимых предметов действия практически идентичны:

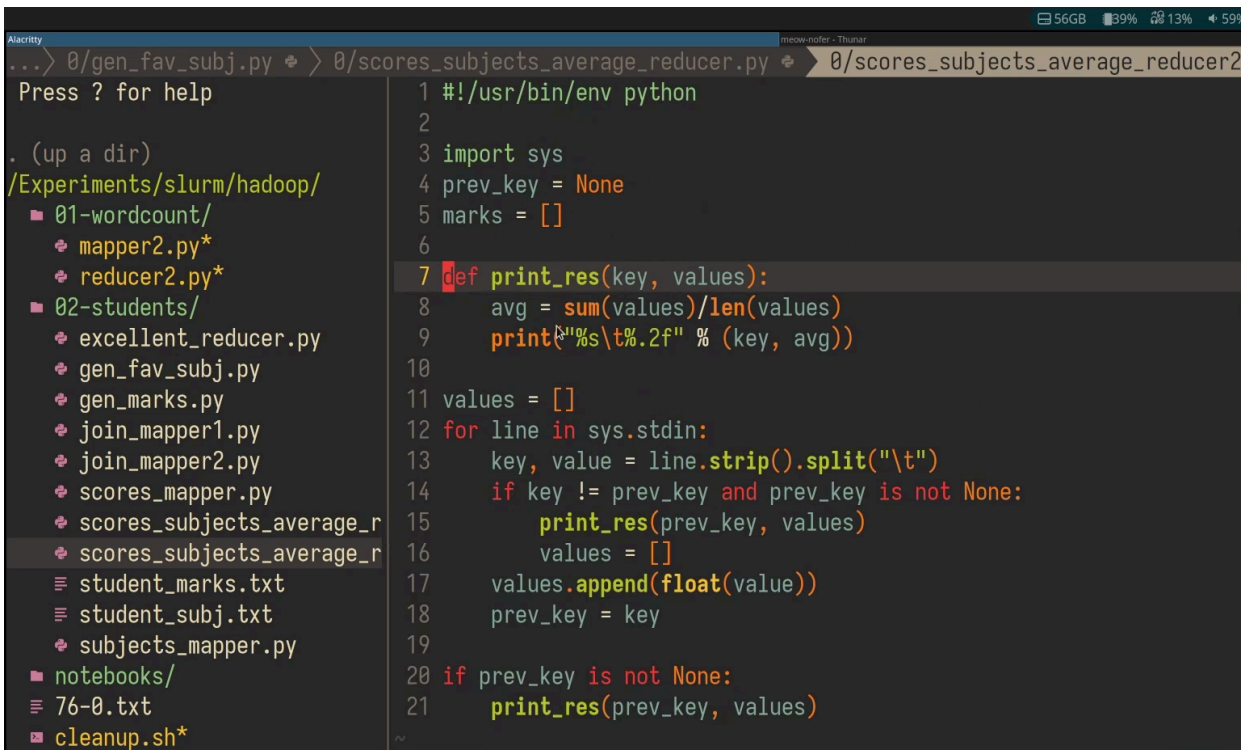


Мы делаем маппер, в котором формат данных такой же, но тип данных будет другой (будет «discipline»). Таким образом, при чтении в job’ах мы будем знать, какие данные из «score», а какие – из «discipline».

На входе все данные будут отсортированы. В reducer считаем среднюю оценку по всем студентам, далее проводим join на стороне reduce:



Мы проходимся по данным, которые поступили нам в редюсере. Если данные «score» и «discipline», то записываем и сохраняем. В данной ситуации мы считаем среднюю оценку (суммируем по score, считаем среднее и возвращаем). Далее, высчитываем среднее по любимому предмету. Мы делаем еще один reduce и в нем делим одно на другое:



Подытожим:

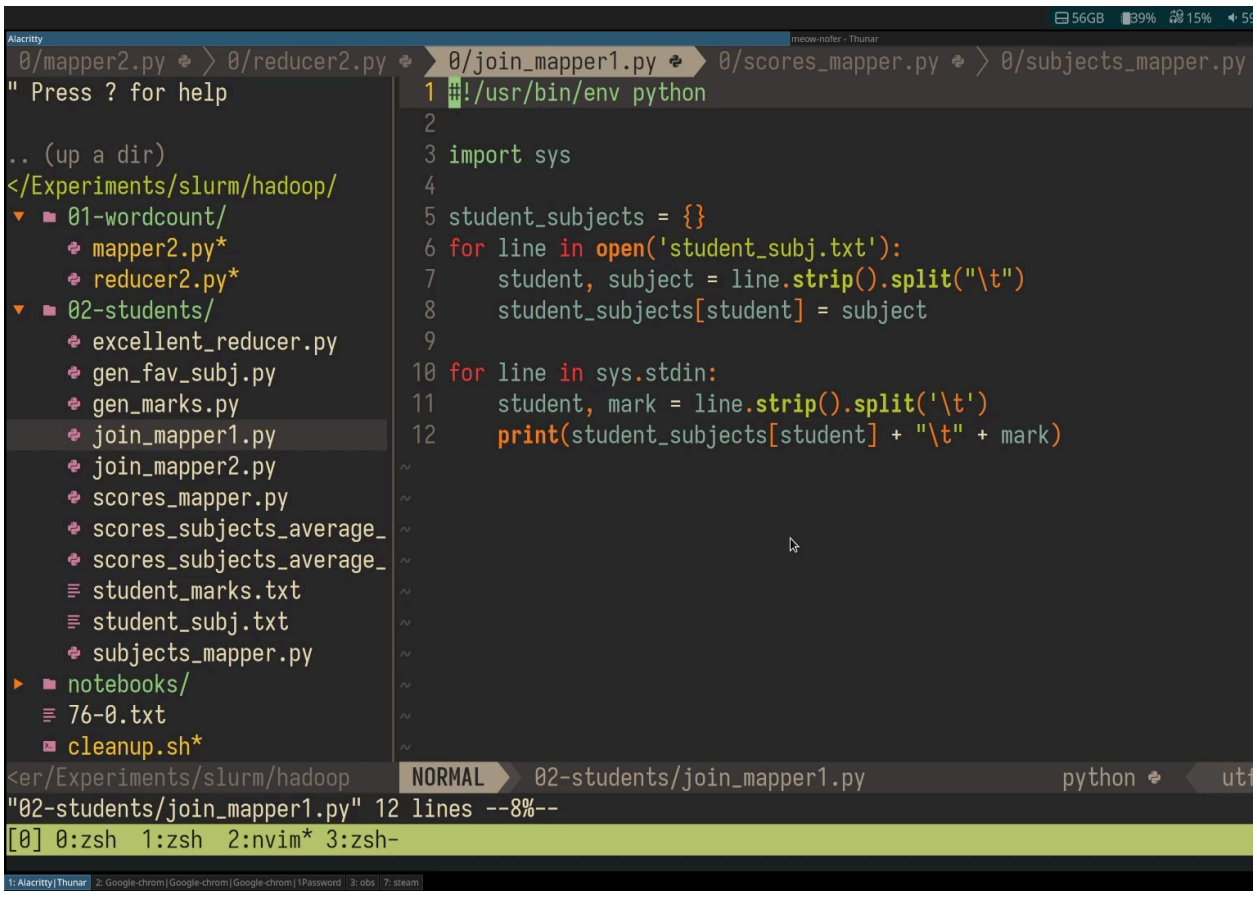
Мы учитываем факт того, что данные сортируются по ключам. Если типы данных разные, то можно их обогатить дополнительной колонкой с этим же типом данных, в reduce сделать дополнительно join.

MapJoin

Довольно часто датасет, с которым мы джойним, небольшой (например, справочник в несколько гб). Мы можем его целиком залить в память на worker’ах и там джойнить.

Так, например, мы считаем среднюю оценку без усреднения по людям. Мы берем список любимых предметов и заливаем целиком, заджойнив с тем, что лежит в памяти, не стакая reduce job’ы.

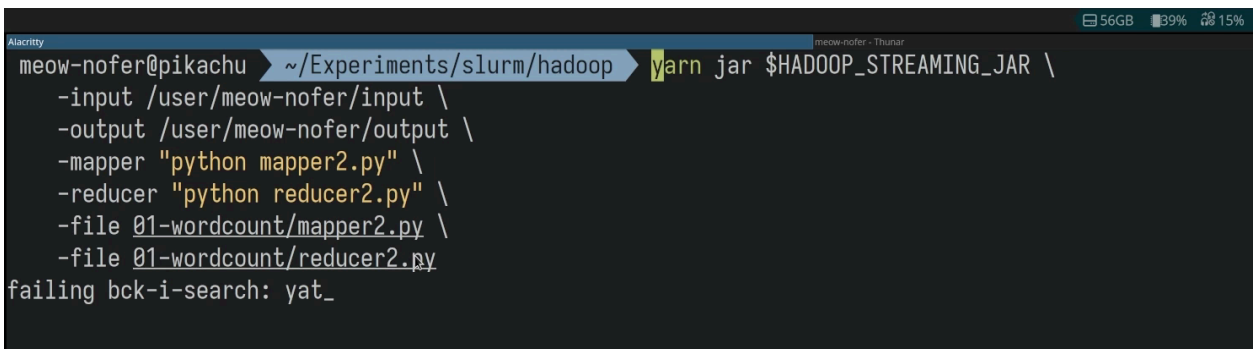
Выглядеть это может следующим образом:



Мы читаем файл, сохраняем информацию о том, у кого какой любимый предмет. Далее, читаем данные и автоматом достаем, что хотим.

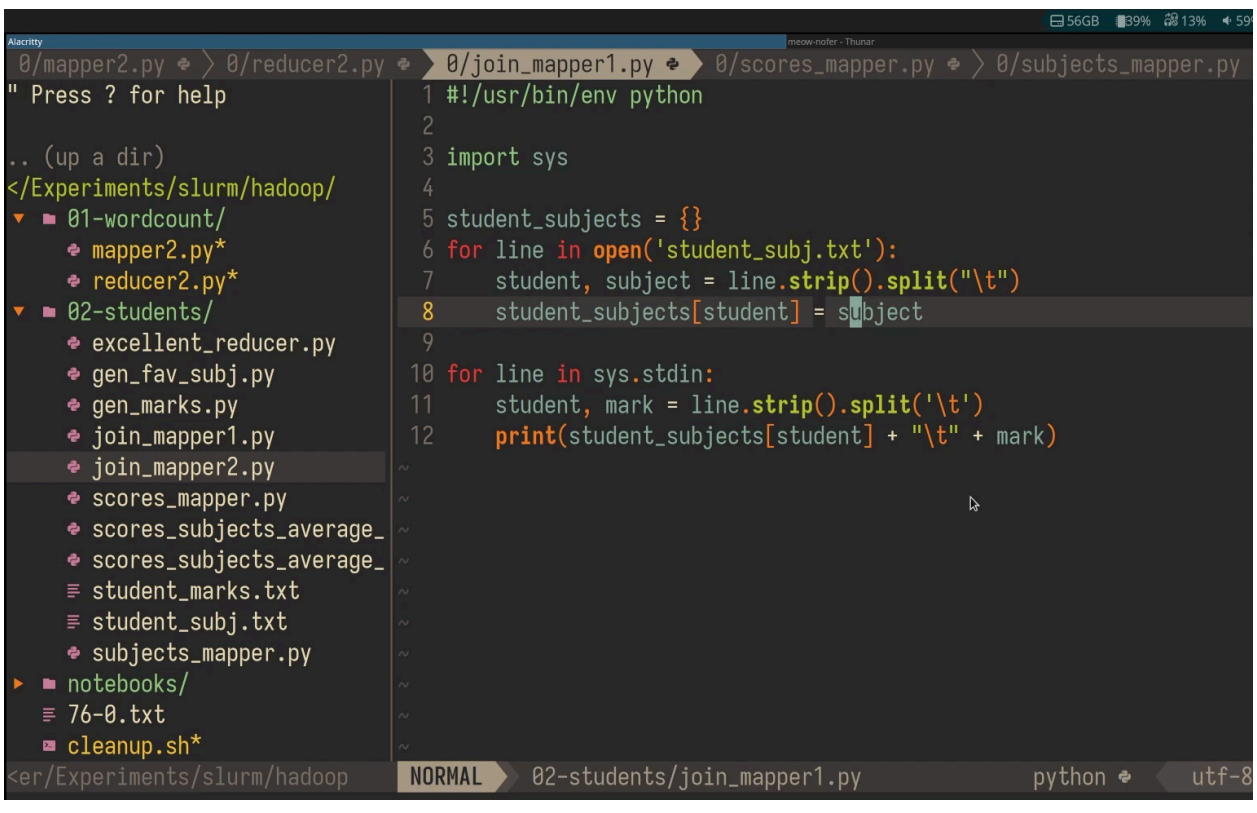
Вопрос: как прочитать данный файл на всех worker'ах, чтобы он был доступен?

Ответ: вспомним синтаксис запуска команд:



У нас есть ключ (files). Используем distributed cache. Более того, файл с данными, который хотим переслать, мы можем заархивировать, а в мапперах читать его из архива.

Так, мы можем указать через файл список любимых предметов. Он зальётся в distributed cache и будет доступен на каждом воркере локально. Таким образом мы сделаем join на этапе маппера:



Важно: это работает только тогда, когда у нас датасет, с которым мы джойним достаточно маленький для того, чтобы поместился в памяти целиком.

Вопрос: зачем нам об этом знать?

Ответ: данная концепция универсальна. Это основополагающие положения, о которых необходимо знать.

Как вам урок? 😠 😐 🔥

Изучил, далее ›