Coroutines Pipeline

> GitHub-classroom для самостоятельно изучающих курс **не** в рамках университетских программ

В этом задании мы пишем аналог bash pipeline, что-то вроде:

```
cat log.txt | grep "ERROR" | awk '{print $2}' | sort | uniq -c
```

В таком pipeline stdout одной программы передаётся как stdin в другую программу

Роль pipe будут выполнять каналы, которые передают данные из одной функции в другую. Само задание по сути состоит из двух частей

1. Написание класс PipelineExecutor с методом execute который обеспечивает конвейерную обработку job-функций (аналог cat, grep, awk, ...)
2. Написание нескольких функций, которые считают нам какую-то условную хеш-сумму от входных данных

Расчет хеш-суммы реализован следующей цепочкой:

- singleHash считает значение crc32(data)+“~”+crc32(md5(data)) (конкатенация двух строк через ~), где data - то что пришло на вход (по сути - числа из первой функции)
- multiHash считает значение crc32(th+data)) (конкатенация цифры, приведённой к строке и строки), где th=0..5 (т.е. 6 хешей на каждое входящее значение), потом берёт конкатенацию результатов в порядке расчета (0..5), где data - то что пришло на вход (и ушло на выход из singleHash)
- combineResults получает все результаты, сортирует, объединяет отсортированный результат через _ (символ подчеркивания) в одну строку
- crc32 считается через функцию dataSignerCrc32
- md5 считается через dataSignerMd5

В чем подвох:

- dataSignerMd5 может одновременно вызываться только 1 раз, считается 10 мс. Если одновременно запустится несколько - будет перегрев на 1 сек
- dataSignerCrc32, считается 1 сек
- На все расчеты у нас 3 сек.
- Если делать в лоб, линейно - для 7 элементов это займёт почти 57 секунд, следовательно надо это как-то распараллелить

From:
http://wiki.osll.ru/ - **Open Source & Linux Lab**

Permanent link:
http://wiki.osll.ru/doku.php/courses:high_performance_computing:coroutines_pipeline

Last update: **2022/11/11 00:49**