Университет ИТМО, факультет программной инженерии и компьютерной техники
Двухнедельная отчётная работа по «Информатике»: аннотация к статье
Дата прошедшей лекции: 11.10.2023 Номер прошедшей лекции: №3 Дата сдачи: 25.10.2023

Выполнил(а) Тимошкин Р.В., № группы P3131, оценка _____
Фамилия И.О. студента не заполнять

Название статьи/главы книги/видеолекции
Python (+numba) быстрее Си — серьёзно?!

ФИО автора статьи (или e-mail)	**Дата публикации (не старше 2020 года)**	**Размер статьи (от 400 слов)**
@axi1	"16" января 2023 г.	2034

Прямая полная ссылка на источник или сокращённая ссылка (bit.ly, tr.im и т. п.)
https://habr.com/ru/articles/484136/
https://habr.com/ru/articles/484142/

Теги, ключевые слова или словосочетания
Python, numba, JIT, LLVM

Перечень фактов, упомянутых в статье (минимум три пункта)
1. Практически сразу после появления Python стали появляться решения для его ускорения.
2. На данный момент самые востребованные — Cython, pypy, numba.
3. numba поддерживает NumPy.
4. В numba своя реализация типизированного словаря. Все ключи должны быть одного типа, ровно как и значения. Питоновский dict нельзя передать в numba, зато нумбовский numba.typed.Dict можно создавать в питоне и передавать в/из нумбы (при этом в питоне он работает чуть медленнее питоновского).
5. Некоторые задачи (например, умножение матрицы на число) распараллеливаются естественным образом.

Позитивные следствия и/или достоинства описанной в статье технологии (минимум три пункта)
1. В numba разгоняется не вся программа, а отдельные функции (с помощью декоратора), это позволяет совместить высокую скорость и обратную совместимость с библиотеками, которые numba (пока) не поддерживает.
2. numba обгоняет Cython за счёт использования нативных процессорных инструкций (Cython не умеет в JIT-компиляцию), а pypy – за счёт более эффективного выполнения байткода LLVM.
3. Сигнатуры для подлежащих компиляции функций (и их локальных переменных!) можно задавать заранее, что позволяет им выполнятся быстро без «прогрева» (первичной генерации LLVM байткода).
4. Numba умеет выполнять разогнанный код на GPU, причём в отличие от того же, например, pycuda или pytorch, не только на nvidia, но и на amd'шных карточках.

Негативные следствия и/или недостатки описанной в статье технологии (минимум три пункта)
1. Никакие другие библиотеки (в частности, scipy и pandas) numba не понимает совсем.
2. Из разогнанных функций можно вызывать только разогнанные, не разогнанные нельзя (хотя разогнанные функции можно вызывать и из разогнанных и из не разогнанных).
3. В разогнанных функциях глобальные переменные становятся константами: их значение фиксируется на момент компиляции функции.
4. Поддержка классов эксперементальна, поэтому ООП код сделать быстрым не получится.
5. Numba до сих пор не хватает толковой документации. Она есть, но в ней есть не всё.

Ваши замечания, пожелания преподавателю *или* анекдот о программистах

```
def f():
      print('Traceback (most recent call last):\n File "<stdin>", line 1, in <module>\n
File "<stdin>", line 3, in f\n File "<stdin>", line 3, in f\n File "<stdin>", line 3, in
f\n [Previous line repeated 992 more times]\n File "<stdin>", line 2, in f\
nRecursionError: maximum recursion depth exceeded while calling a Python object')
    f()
f()
```