Ускорение программ на Python с помощью Cython и numba

Москаленко Роман Борисович

May 2020

1 Введение

В проекте по моделированию пространственно-распределённых игр основной задачей являлось написание кода для моделирования игры. Мы моделируем игру Новака-Мея, основанную на дилемме заключённого. В оригинальном варианте игра происходит на плоском квадратном поле, в каждой клетке поля расположены агенты. Все агенты следуют одной из двух стратегий: **C** - агент сотрудничает со всеми, **D** - агент отказывается сотрудничать. На каждом шаге игры все агенты играют со своими соседями и с собой (всего 9 игр для каждого человека), агенты, находящиеся у края поля играют с агентами с другой стороны поля. Агенты получают очки согласно таблице.

стратегия		счёт	
агент №1	агент №2	агент №1	агент №2
C	**C**	1	1
C	**D**	0	**b**
D	**D**	0	0

Параметр **b** задаётся в начале игры и определяет её ход. Когда все пары агентов сыграют, каждый агент меняет свою стратегию на стратегию соседа с наибольшим счётом. Затем все шаги повторяются.

В нашем случае игра происходит на простой трёхмерной кубической решётке (у каждого агента 27 соседей, включая его самого).

2 Код

2.1 Базовая функция

Базовая версия основной функции была написана на чистом питоне. Выполнение одного шага игры у этой версии функции на поле размером $60 \times 60 \times 60$ занимает примерно 16 секунд. Учитывая, что для замеров требовалось сделать более 10000 шагов на 1500 различных полях, это могло занять несколько месяцев. Для уменьшения времени работы функции я использовал Cython и Numba.

Все замеры времени проводились на суперкомпьютерном кластере Высшей Школы Экономики. В репозитории 1 находится код.

2.2 Cython

Cython преобразует код написанный на Python в код си, и затем позволяет компилировать его и использовать коде на Python. Основным преимуществом данного метода является строгое определение типов переменных на уровне си, что значительно ускоряет выполнение операций с этими переменными.

Код главной функции на Cython отличается от предыдущей версии только строго определёнными типами переменных. Время работы данной функции на поле $60 \times 60 \times 60$ 28 миллисекунд, что примерно в 600 раз быстрее версии на чистом питоне.

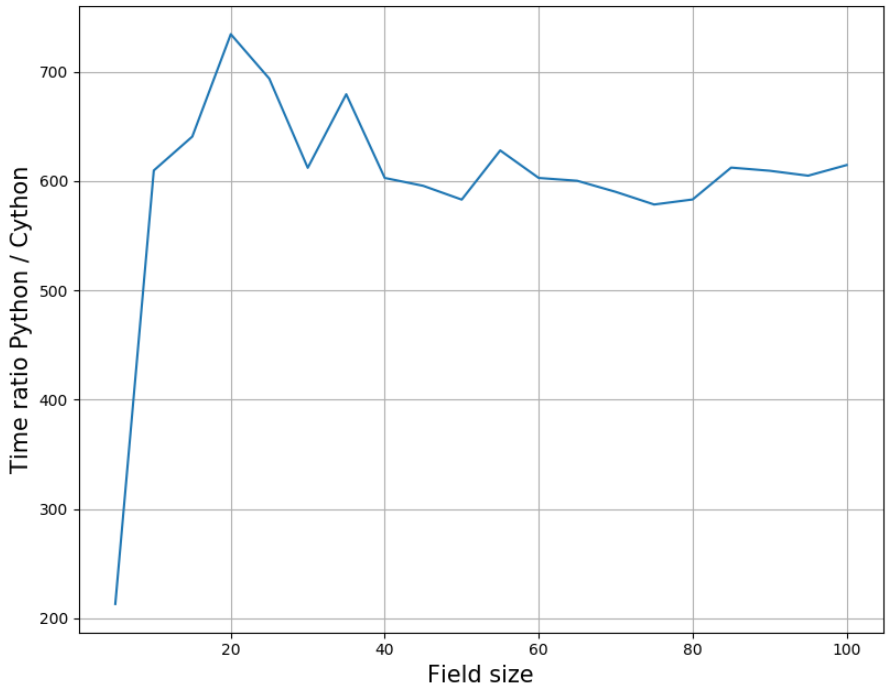

Рис. 1. Отношение времени работы функции на чистом Python и с Cython

Стоит отметить, что при разработке в Jupyter notebook действительно было достаточно определить типы переменных и обернуть функцию в декоратор. Но для замеров на кластере было необходимо написать код запускаемы на питоне без каких-либо оболочек. В таком случае декораторы, предоставляемые Jupyter notebook использовать не получится, и функцию придётся компилировать на си с помощью отдельно написанного подготовительного скрипта и потом вызывать как библиотеку в основном коде. Помимо того, что подобные действия не требуются при использовании компилятора numba (который будет описан ниже), компиляция происходит под текущую платформу, из-за чего мне пришлось отдельно делать это на моём компьютере и на кластере, так как первый работает на Windows, а второй на Linux.

2.3 Numba

Numba это компилятор на лету, основан на технологии LLVM. Он компилирует байт-кода питона в машинный код, за счёт чего и происходит

ускорение.

Версия с Numba состоит из подготовительной функции, в которой заранее создаются нулевые массивы, и массив с игровым полем из трёхмерного переводится в одномерный, и основной функции, которая вызывается из подготовительной. Основная функция почти не отличается от функции на чистом питоне, но обёрнута в декоратор. Подготовительная функция нужна, чтобы не использовать numpy в обёртываемой функции, так как numba поддерживает не все операции из данной библиотеки. Время работы 36 миллисекунд

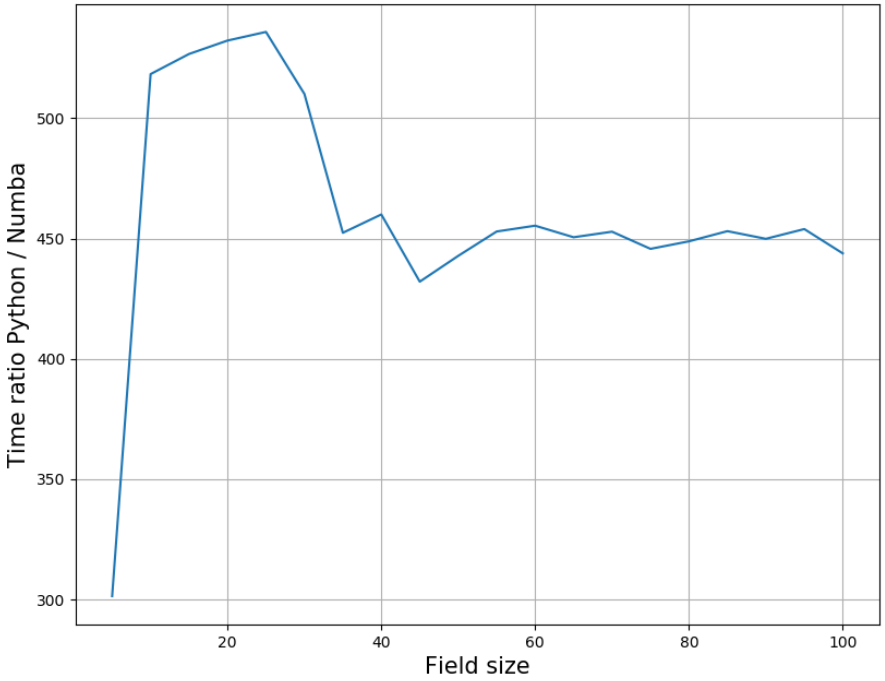

Рис. 2. Отношение времени работы функции на чистом Python и с Numba

3 Итог

Оба метода значительно ускоряют код. Несмотря на то, что Cython быстрее чем numba, отношение времени работы порядка единицы, при этом переделать функции под Numba значительно проще, особенно если в ней не используются другие библиотеки, нужно буквально написать од-

ну строчку. Numba хорошо работает на разных платформах, в то время как функцию с Cython мне приходилось отдельно компилировать для Windows на своём компьютере и для Linux на кластере. Использовать numba гораздо проще и удобнее, если не требуется предельно высокая скорость работы, данный метод является отличной альтернативой Cython.

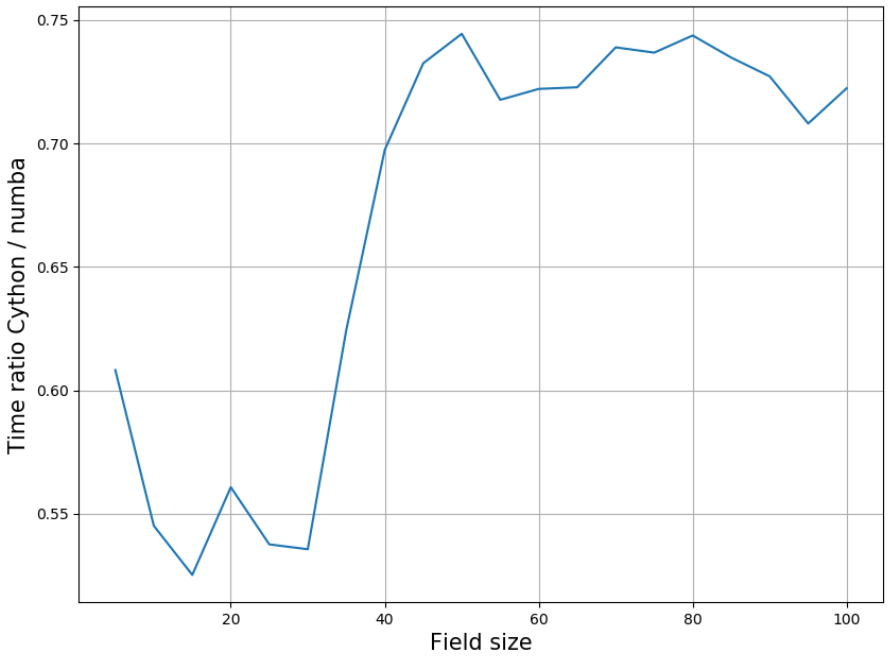

Рис. 3. Отношение времени работы функции с Cython и с Numba

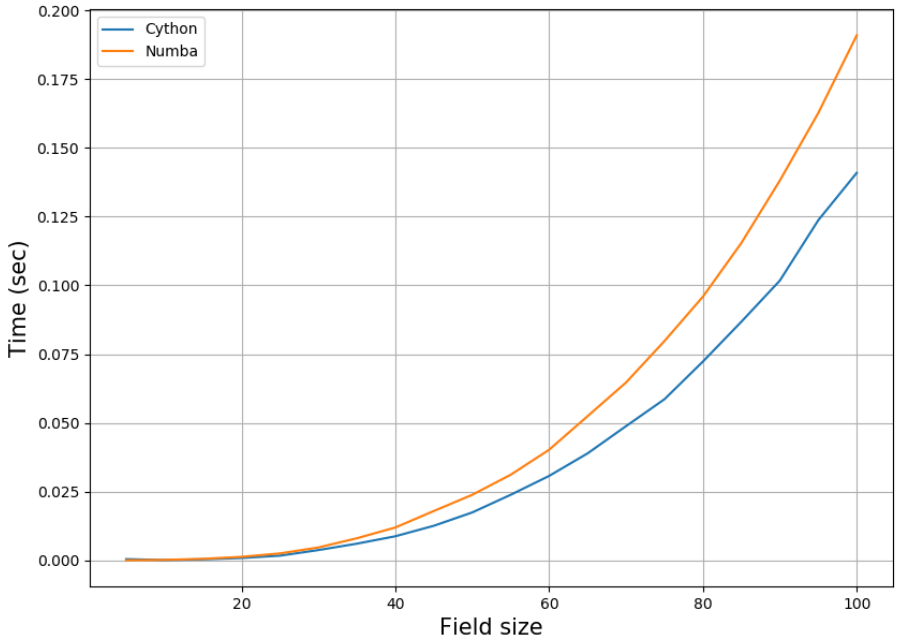

Рис. 4. Время работы функций с Cython и с Numba

Список литературы

[1] Репозиторий github