УДК 004.43
ББК 32.973.26-018.1
Г68

Горелик, Миша.

Г68 Высокопроизводительные Python-приложения. Практическое руководство по эффективному программированию / Миша Горелик, Йен Освальд ; [перевод с английского М. А. Райтман]. — Москва : Эксмо, 2022. — 528 с. — (Мировой компьютерный бестселлер).

ISBN 978-5-04-113372-6

Эта книга ориентирована на профессиональных Python-программистов, разрабатывающих приложения, задействующие большие объемы данных. Она расскажет, как избежать узких мест в коде, сделать его работу максимально эффективной и разрабатывать высокопроизводительные системы. Авторы знакомят читателей с инструментами профилирования и компиляции и рассказывают, как их использовать, разбирают вопросы взаимодействия Python с архитектурой ПК и дают множество важных и эксклюзивных рекомендаций для разработчиков.

УДК 004.43
ББК 32.973.26-018.1

ISBN 978-5-04-113372-6

Оглавление

Вступление

В контексте разговора о высокопроизводительных вычислениях вы наверняка представляете себе гигантские кластеры машин, моделирующих сложные погодные явления или пытающихся вычленить сигналы из данных, полученных от далеких звезд. Вполне очевидно предположить, что о производительности кода думают только люди, создающие такие специализированные системы. Однако открыв данную книгу, вы уже сделали шаг к освоению теории и практики, которые помогут вам написать высокопроизводительный код. Согласитесь, умение создавать высокопроизводительные системы может быть чрезвычайно полезным для каждого программиста.

Всем нам знаком определенный ряд приложений, реализованных, скажем так, на грани возможного, и приблизиться к ним, не написав оптимальный по производительности код, не получится. Если вы сейчас работаете над чем-то подобным, вы попали по адресу. Ведь пул приложений, которым был бы полезен высокопроизводительный код, на самом деле очень и очень широк.

Принято считать, что основным драйвером инноваций являются новые технические возможности, однако я выступаю также и за возможности, которые на порядки увеличивают доступность технологий. Когда на что-то начинает тратиться в десять раз меньше времени или вычислительных ресурсов, внезапно набор приложений, которые вы можете использовать, становится гораздо шире, чем вы себе представляли.

Впервые с данным принципом на личном опыте я столкнулась более десяти лет назад, когда, работая в социальных медиа, провела вместе с коллегами анализ нескольких терабайт данных, чтобы определить, по чьим фотографиям — кошек или собак — люди в соцсетях кликают чаще.

Конечно, победили собаки. У кошек просто рекламы больше.

На тот момент подобное исследование было исключительно легкомысленной тратой вычислительного времени и ресурсов! Появление возможности применять методы, ранее доступные только для чрезвычайно важных задач, таких как обнаружение мошенничества, для решения, казалось бы, тривиального вопроса открыло удивительные перспективы. Мы смогли извлечь уроки из этих экспериментов и создать совершенно новый набор продуктов для поиска и обнаружения контента.

В качестве примера, который вы можете увидеть и в жизни, рассмотрим систему машинного обучения для распознавания животных и людей на видеозаписях с камер видеонаблюдения. Достаточно производительная система может позволить вам встроить данную опцию в саму камеру, что, конечно, положительно отразится на безопасности, или в облачный ресурс, что приведет к снижению потребления вычислительных ресурсов и энергии, а значит, и эксплуатационных затрат. Это в свою очередь позволит высвободить ресурсы для рассмотрения смежных проблем, соответственно, потенциально приведет к росту ценности системы.

Разумеется, все мы хотим создавать системы, являющиеся одновременно эффективными, простыми для понимания и производительными. К сожалению, нам часто приходится делать выбор в пользу только двух (а иногда и одного) параметров из трех! Книга, которую вы держите сейчас в руках, предназначена для тех, кто хочет выбирать все три!

Есть три вещи, выделяющие данное руководство на фоне остальных. Во-первых, оно разработано для нас — людей, которые пишут код. В книге изложена вся необходимая информация, позволяющая понять, почему вы можете сделать тот или иной выбор. Во-вторых, авторы являются прекрасными кураторами и превосходно объясняют теорию. Наконец, в-третьих, в данном обновленном издании вы познакомитесь с особенностями наиболее полезных современных библиотек для реализации описанных подходов.

Это одна из тех редких книг, что способна буквально перевернуть ваше представление о практическом программировании. Я давала ее многим людям, которым могут пригодиться предложенные в ней инструменты. Идеи, описанные на страницах книги, позволят вам поднять свой профессиональный уровень, независимо от того, на каком языке или в какой среде вы работаете.

Наслаждайтесь!

Хилари Мейсон,
Дата-саентист в Residence at Accel

Предисловие

Изучить Python легко. Однако вы здесь, скорее всего, из-за того, что ваш код работает верно, но недостаточно эффективно. Безусловно, приятно, когда код легко поддается изменениям и вы можете быстро перебирать идеи. Тем не менее компромисс между *простотой разработки* и *максимальной эффективностью* — это вполне понятный, пусть и часто огорчающий, феномен. Но решение существует.

Кто-то сталкивается с задачами, имеющими последовательные процессы, которые должны выполняться быстрее. Или с проблемами, решаемыми за счет использования многоядерных архитектур, кластеров или графических процессоров. Некоторым нужны масштабируемые системы, способные без потери надежности обрабатывать больше или меньше данных, насколько позволяют целесообразность и средства. Кто-то сталкивается с тем, что его методы кодирования, часто заимствованные из других языков, возможно, не так естественны, как примеры, которые он видит у других.

В данной книге мы рассмотрим все эти темы, приведем практические рекомендации по выявлению узких мест и созданию более эффективных и масштабируемых решений. Мы также включили в нее несколько «историй с передовой» от тех, кто первым столкнулся с подобными проблемами и проторил для вас путь.

Python хорошо подходит для быстрой разработки, производственного развертывания и создания масштабируемых систем. В его экосистеме полно людей, которые работают над ее масштабированием, и это дает вам больше времени на более сложные задачи.

Для кого предназначена эта книга

Вы, вероятно, имели дело с Python достаточно долго, чтобы понимать, почему некоторые вещи в нем работают медленно, и видели такие технологии, как Cython, numpy и PyPy, которые часто обсуждаются в контексте вопроса увеличения скорости приложений. Возможно, вы также программировали на других языках и знаете, что проблему производительности можно решить далеко не одним-единственным способом.

Хотя данная книга в первую очередь предназначена тем, кто сталкивается с проблемами, касающимися эксплуатации процессора, в ней также рассмотрены вопросы передачи данных и решения, связанные с оперативной памятью. Обычно с подобными вещами сталкиваются ученые, инженеры и академики.

Помимо прочего мы рассмотрим вопросы, актуальные для веб-разработчиков, например, перемещение данных и использование JIT-компиляторов вроде PyPy и асинхронного ввода-вывода, для увеличения производительности.

Вам может помочь (но это вовсе не обязательное условие) наличие опыта работы с C (или C ++, или, возможно, Java). Самый распространенный интерпретатор Python (CPython — тот самый, который запускается, когда вы набираете в командной строке команду `python`) написан на C, поэтому все его хуки и библиотеки работают со сложными внутренними механизмами C. Однако мы рассмотрим и множество других техник, которые вообще не предполагают знания C.

Вам также могут пригодиться базовые знания о работе ЦП, об архитектуре памяти и шинах данных, но, опять же, это не обязательно.

Для кого эта книга не предназначена

Данная книга предназначена для программистов на Python от среднего до продвинутого уровня. Заинтересованные начинающие программисты тоже могут почитать, но мы рекомендуем сначала как следует изучить Python.

Мы не будем рассматривать вопросы оптимизации системы хранения. Если ваши проекты касаются SQL или NoSQL, данная книга, вероятно, вам не поможет.

Что вы узнаете

У нас, авторов книги, есть опыт работы с большими объемами данных, остро реагирующими на производительность приложений, а также многолетняя потребность в масштабируемых архитектурах в корпоративной и учебной деятельности. Мы постараемся поделиться нашим трудно заработанным опытом, чтобы спасти вас от ошибок, которые сами совершали в свое время.

В начале каждой главы мы перечисляем вопросы, на которые далее отвечаем (если вы вдруг не находите для себя ответа, сообщите нам, и мы исправим недочеты в следующем издании!).

В целом мы предлагаем обсудить следующие темы:

— Оборудование компьютера, чтобы вы знали, как все функционирует.
— Списки и кортежи — тонкие различия в семантике и производительности в этих структурах данных.
— Словари и множества — стратегии выделения памяти и алгоритмы доступа в этих структурах данных.
— Итераторы — как писать на Python и обрабатывать бесконечные потоки данных с помощью итераций.
— Эффективное использование Python и его модулей.
— Матрицы `numpy` — как выжать максимум из вашей любимой библиотеки `numpy`.
— Компиляция и своевременные вычисления — обработка выполняется эффективнее за счет компиляции в машинный код, гарантируя, что были достигнуты результаты профилирования.
— Параллельная обработка — способы эффективно перемещать данные.
— `multiprocessing` — различные способы применения встроенной библиотеки `multiprocessing` для организации параллельных вычислений, эффективного совместного использования матриц `numpy`, а также некоторые недостатки и преимущества межпроцессного взаимодействия (IPC).
— Кластерные вычисления — готовим код из `multiprocessing` для работы в локальном или удаленном кластере как в исследовательских, так и в производственных системах.
— Оптимизация использования оперативной памяти — подход к решению серьезных задач без покупки огромного компьютера.
— Истории с фронта — уроки, изложенные в виде рассказов тех, кто бросился на амбразуру новых проблем.

Python 3

Версия Python 3 стандатизирована в 2020 году, а Python 2.7 после 10-летнего процесса миграции объявлен устаревшим. Если вы все еще используете Python 2.7, то зря — многие библиотеки уже не поддерживаются этой версией, и со временем поддержка прекратится вовсе. Пожалуйста, сделайте одолжение сообществу, перейдите на Python 3 и создавайте все ваши новые проекты на нем.

В данной книге мы используем 64-разрядную версию Python. 32-разрядная версия тоже подойдет, однако она гораздо менее распространена в сфере разработки приложений для работы с большими объемами данных. В случае с ней все

библиотеки будут работать как обычно, но вычислительная точность, которая зависит от разрядности системы, скорее всего, изменится. Доминирующей в нашей области является 64-разрядная версия, наряду со средами *nix (чаще Linux или Mac). Дело в том, что 64-разрядная версия поддерживает бóльшие объемы модулей памяти ОЗУ, в то время как среда *nix позволяет создавать приложения, обладающие хорошо понятными способами развертывания и настройки и понятным поведением.

Если вы являетесь пользователем Windows, придется вас немного огорчить. Бóльшая часть приведенных в книге листингов будет работать нормально, но есть и специфические для ОС вещи, которые потребуют от вас самостоятельного изучения и корректировки под Windows. Самая большая трудность, с которой рискует столкнуться пользователь Windows, — это установка модулей, однако можно подсмотреть необходимые решения на сайтах вроде StackOverflow. Если вы работаете в Windows, то наличие виртуальной машины (например, VirtualBox) с запущенной Linux поможет вам экспериментировать более свободно.

Пользователям Windows обязательно стоит взглянуть на готовые пакетные решения, подобные тем, что доступны через Anaconda, Canopy, Python(x, y) или Sage. Данные дистрибутивы сильно упростят жизнь также и пользователям Linux и macOS.

Отличия от Python 2.7

После перехода с Python 2.7 необходимо помнить о некоторых важных изменениях:

— Операция «/», означающая в Python 2.7 *целочисленное* деление, в Python 3 представляет собой деление с *плавающей точкой*.
— Для представления текстовых данных в Python 2.7 применялись типы `str` и `unicode`, а в Python 3 все строки имеют тип `str`, и это всегда Unicode. Для ясности используется тип `bytes`, который позволяет обрабатывать некодированные байтовые последовательности.

Если вы как раз переводите ваш код на Python 3, рекомендуем два хороших руководства: *Porting Python 2 Code to Python 3*[1] и *Porting to Python 3: An in-depth*

[1] **bit.ly/pyporting**

guide[1]. Имея дистрибутив вроде Anaconda или Canopy, вы сможете запускать Python 2 и Python 3 одновременно — это упростит перенос.

Лицензия

Данная книга лицензирована с помощью Creative Commons Attribution-NonCommercial-NoDerivs3.0[2].

Вы можете использовать ее в некоммерческих целях, в том числе для некоммерческого обучения. Лицензия допускает только полное воспроизведение, для получения прав на частичное воспроизведение, пожалуйста, свяжитесь с издательством O'Reilly.

Ссылки на книгу

Лицензия Creative Commons требует, чтобы вы ссылались на источник, даже если используете лишь часть книги. В идеале подобная ссылка должна выглядеть так: «Высокопроизводительные Python-приложения (2-е издание), Миша Горелик и Йен Освальд».

Ошибки и отзывы

Мы будем очень признательны вам за отзывы или рецензии на общедоступных сайтах, таких как, например, Amazon — так вы поможете другим людям понять, будет ли им полезна наша книга. Кроме того, вы можете написать нам на адрес электронной почты **feedback@highperformancepython.com**.

Мы очень хотим услышать ваше мнение о книге, не важно, пойдет ли речь о найденных вами ошибках, о полученной пользе или о высокопроизводительных методах, которые стоит описать в следующем издании. Посетить страницу книги на сайте издательства O'Reilly можно по адресу **https://oreil.ly/highperformance-python-2e**.

Жалобы принимаются через службу мгновенной передачи жалоб `> /dev/null`.

[1] **python3porting.com**

[2] **bit.ly/CC_A-NC-ND3**

Условные обозначения

В данной книге используются следующие условные обозначения.

Курсив
: Новые термины, имена файлов и расширения файлов.

Полужирный
: URL-адреса и адреса электронной почты.

Моноширинный текст
: Используется для листингов программ, а также внутри текста для ссылок на элементы программы, такие как имена переменных или функций, базы данных, типы данных, переменные среды, операторы и ключевые слова.

Моноширинный полужирный
: Команды или другой текст, который пользователь должен вводить буквально.

Моноширинный курсив
: Показывает текст, который следует заменить значениями, заданными пользователем, или значениями, определяемыми контекстом.

Элемент, обозначающий совет, предложение или важную информацию для обдумывания.

Элемент, обозначающий общее примечание.

Элемент, указывающий на предупреждение или предостережение.

Использование примеров кода

Дополнительные материалы (примеры кода, упражнения и т. д.) можно скачать по ссылке **github.com/mynameisfiber/high_performance_python_2e**.

Если у вас возникнет технический вопрос или проблема с использованием примеров кода, отправьте электронное письмо на адрес bookquestions@oreilly.com.

Благодарности

Спасибо Хилари Мейсон за замечательное вступительное слово, которое она написала для нашей книги. Джайлз Уивер и Дмитрий Денисенок предоставили бесценные технические отзывы; отличная работа, парни.

Благодарим Патрика Купера, Кирана Дейла, Дэна Форман-Макки, Кэлвина Джайлса, Брайана Грейнджера, Хилари Мейсон, Джейми Мэтьюза, Джона Монтгомери, Кристиана Скоу Оксвига, Мэтта «snakes» Рейферсона, Бальтазара Руберола, Майкла Скирпана, Люка Андервуда, Джейка Вандерпласа и Уильяма Винтера за неоценимый вклад в создание книги.

Йен благодарит свою жену Эмили за то, что позволила ему на восемь месяцев выпасть из жизни во время работы над книгой (к счастью, она все понимает). Йен также просит прощения у своей собаки за то, что корпел над рукописью и мог выгуливать ее в лесу меньше, чем ей бы того хотелось.

Миша благодарит Мэрион, своих друзей и семью за терпеливое и чуткое отношение к его работе над книгой.

Спасибо сотрудникам издательства O'Reilly, с которыми так приятно работать.

Авторы главы 12 любезно поделились своим временем и опытом. Мы благодарим Соледад Галли, Линду Уручурту, Ванентина Ханела и Винсента Д. Вармердама за это издание, а также Бена Джексона, Радима Жегуржека, Себастьяна Трепку, Алекса Келли, Марко Ташича и Эндрю Годвина за их время и усердие в работе над предыдущим изданием.

ГЛАВА 1

Общие понятия о высокой производительности в Python

Вопросы, которые мы рассмотрим в данной главе:

— Каковы основные элементы архитектуры компьютера?
— Какие альтернативные варианты архитектуры более распространены?
— Как Python абстрагирует базовую компьютерную архитектуру?
— Какие препятствия могут встретиться на пути к созданию производительного кода Python?
— Какие стратегии помогут вам стать разработчиком высокопроизводительных приложений?

В целом программирование можно рассматривать как перемещение и разного рода преобразование данных для достижения определенного результата. Разумеется, эти операции требуют времени. Следовательно, *высокопроизводительное программирование* можно рассматривать как меры по минимизации временных затрат либо за счет сокращения трудозатрат (т. е. написания более эффективного кода), либо за счет выбора оптимального способа выполнения этих операций (т. е. поиска более подходящего алгоритма).

Мы предлагаем в первую очередь сосредоточиться на сокращении трудозатрат, чтобы лучше понять актуальное оборудование, используемое в работе. Это занятие может показаться бесполеным, поскольку Python, напротив, предлагает абстрагироваться от непосредственного взаимодействия с аппаратным обеспечением. Однако понимание того, как лучше работать с данными на реальном оборудовании и как именно Python выполняет эти операции, поможет вам продвинуться в написании высокопроизводительных программ на Python.

Базовая компьютерная система

Основные компоненты, из которых состоит компьютер, можно условно отнести к трем общим группам: вычислительные блоки, блоки памяти и соединения