Оглавление

Предисловие

Несколько лет назад один из процессов на основе Python, которым занималась моя команда разработчиков, вдруг наглухо подвис. Он продолжал нагружать процессор, но не завершался. Это был один из критически важных процессов для компании, и нам необходимо было срочно разобраться с возникшей ситуацией. Мы взглянули на алгоритм и не обнаружили каких-то серьезных проблем. Да там и не было ничего сложного. После нескольких часов работы мы поняли, что узким местом является процесс поиска в довольно объемном списке. Заменив список на множество, мы решили проблему. По сути, мы изменили структуру для хранения данных, тем самым снизив время поиска с нескольких часов до миллисекунд.

Это было какое-то прозрение, которое заставило меня задуматься о том, что:

- проблема была пустяковой, но с помощью нее мы выявили, что в процессе разработки совершенно не заботились об эффективности своего кода. К примеру, если бы мы в своей работе пользовались профайлером, то смогли бы обнаруживать подобные утечки за минуты, а не тратить на это часы;
- в выигрыше в итоге остались все: мы одновременно сократили время выполнения процесса и снизили объем используемых ресурсов памяти. Да, зачастую в подобных ситуациях мы вынуждены идти на компромиссы, но здесь мы имели дело с беспроигрышной партией;
- в глобальном смысле также никто не остался в накладе. Во-первых, ускорение процесса пошло на пользу компании, а во-вторых, решение проблемы позволило снизить процессорное время, что положительно сказывается на потреблении электричества и экологии в целом;

хотя в этом конкретном случае о большой экономии ресурсов речи не шло, кому-то наверняка приходится ежедневно сталкиваться с подобными ситуациями.

В итоге я решил посвятить свое время написанию книги, с помощью которой смогу транслировать свои озарения другим программистам. Моя главная миссия – помочь бывалым программистам на языке Python разрабатывать и внедрять более эффективный код и легко распознавать ситуации, в которых можно и нужно идти на компромиссы. При этом я постарался подойти к проблеме комплексно, рассмотрев как базовый Python, так и основные библиотеки и уделив внимание алгоритмам, архитектуре современного аппаратного обеспечения и эффективности процессоров и способов хранения данных. Надеюсь, книга, которую вы держите в руках, поможет вам более уверенно подходить к решению проблем, связанных с производительностью, в экосистеме Python.

О книге

Целью создания этой книги в первую очередь была помощь программистам в написании высокоэффективных приложений в рамках экосистемы Python. Под эффективностью приложений я прежде всего подразумеваю снижение процессорного времени, а также расходования памяти и сетевых ресурсов при их выполнении.

В книге мы будем затрагивать все аспекты разработки приложений, так или иначе относящиеся к производительности. При этом мы не будем ограничивать себя оптимизацией только в базовом функционале Python, а рассмотрим приемы эффективного использования популярных библиотек, таких как NumPy и pandas. Кроме того, поскольку Python не всегда способен похвастаться быстродействием, мы также при необходимости будем обращаться за помощью к более эффективному расширению языка под названием Cython. Помимо этого, мы затронем вопросы влияния аппаратного обеспечения на эффективность кода. В частности, проанализируем взаимосвязь между современной архитектурой компьютеров и быстродействием выполняемых алгоритмов. Также мы изучим воздействие на производительность конфигурации сети и рассмотрим возможность использования вычислительных ресурсов графического процессора (GPU) для быстрого анализа данных.

Для кого эта книга

Эта книга рассчитана на программистов с определенным опытом. Читая содержание книги, вы должны быть более или менее знакомы с большинством упоминающихся в нем технологий. А если вам довелось поработать с какими-то из них, вообще прекрасно. За исключением разделов, посвященных библиотекам ввода-вывода и вычислениям с помощью GPU, мы не будем сильно вдаваться в описание базовых вещей, а будем полагаться на то, что вы их и так знаете.

Если в настоящее время вы пишете код и думаете о том, как сделать его максимально эффективным, эта книга точно для вас.

И все же для извлечения максимальной пользы из данной книги вы должны обладать хотя бы двухлетним опытом разработки на языке Python, знать его основные управляющие структуры и понимать, как обращаться со списками, множествами и словарями. У вас также желательно должен быть опыт работы с популярными библиотеками Python, такими как os, sys, pickle и multiprocessing. Кроме того, чтобы воспользоваться всеми преимуществами показанных в этой книге техник, вы должны неплохо ориентироваться в таких популярных пакетах, как NumPy с его массивами и pandas с датафреймами.

Было бы здорово, если бы вы обладали некоторыми знаниями, пусть и не практическими, в области оптимизации кода на Python с привлечением сторонних языков программирования наподобие C или Rust или с использованием других подходов, включающих задействование расширения Cython или компилятора Numba. Практические наработки в области библиотек ввода-вывода в Python также помогут вам в освоении материала этой книги. Поскольку эти библиотеки не так широко освещаются в литературе, мы начнем с самого начала и познакомимся с таким форматом, как Apache Parquet, и пакетом Zarr.

Кроме того, вам необходимо знать основные команды для работы с терминалом Linux (или MacOS). Если у вас Windows, установите любую оболочку на основе Unix или заручитесь необходимыми знаниями для работы с командной строкой или оболочкой PowerShell. Ну и, конечно, без установленного на компьютере интерпретатора Python вам будет не обойтись.

При необходимости я буду давать определенные рекомендации по работе с облачными ресурсами, но доступ к облаку или какие-то особые знания в этой области при чтении книги вам не понадобятся. Если вы заинтересованы в работе с облаком, вы можете узнать все необходимое о приобретении и настройке своей среды у вашего провайдера.

Хотя мы не подразумеваем каких-то особых углубленных знаний с вашей стороны в области оптимизации кода, базовые понятия о скорости выполнения алгоритмов вам не помешают. Например, вы должны понимать, что алгоритмы, масштабирующиеся с ростом объема данных линейно, лучше тех, что масштабируются экспоненциально. Что касается оптимизации вычислений при помощи графического процессора, в этой области вам не потребуется никаких предварительных знаний.

Организация книги

Главы в этой книге по большей части независимы друг от друга, и вы можете перепрыгивать между ними по своему желанию. Несмотря на это, книга состоит из четырех частей.

Часть I (главы 1–4). Фундаментальные подходы. Здесь будет в основном вводный материал:

- в главе 1 мы сформулируем для себя проблему и определимся с тем, зачем именно нужно повышать эффективность в области вычислений и хранения информации. Также в этой главе мы представим целостный подход, применяемый в книге, и обеспечим вас необходимой навигацией;
- глава 2 будет посвящена оптимизации в базовом Python. Мы также коснемся вопросов повышения производительности при использовании структур данных языка Python и поговорим о профилировании кода, выделении памяти и техниках, связанных с ленивыми (отложенными) вычислениями;
- в главе 3 мы будем обсуждать конкурентность и параллелизм в Python, а также узнаем, как можно наиболее эффективно использовать многопроцессную обработку и многопоточность. Одновременно мы затронем вопросы ограничений параллельной обработки при использовании потоков. В этой главе также будет рассмотрена асинхронность как эффективный способ обработки множества конкурентных запросов с низкой нагрузкой, что типично при работе с веб-службами;
- в главе 4 познакомимся с библиотекой NumPy, позволяющей эффективно работать с многомерными массивами. NumPy лежит в основе всех современных техник обработки данных, что делает эту библиотеку одной из ключевых в Python. В этой главе мы затронем специфичные для NumPy техники, позволяющие создавать более эффективный код, такие как представления, транслирование и векторизация.

Часть II (главы 5–6). Аппаратное обеспечение. Здесь мы сосредоточимся на извлечении максимума производительности из имеющихся аппаратных и сетевых ресурсов:

- в главе 5 познакомимся с расширением языка Python под названием Cython, позволяющим создавать гораздо более эффективный код. Python – это интерпретируемый высокоуровневый язык, а значит, от него не стоит ожидать оптимизации под конкретное аппаратное обеспечение. В то же время есть языки программирования, такие как C или Rust, которые способны оптимизировать код под «железо». Cython принадлежит к тому же подмножеству языков, оставаясь при этом в тесном родстве с языком Python. В то же время он позволяет компилировать код Python в C. Для создания эффективного кода на Cython необходимо придерживаться определенных правил в отношении реализации. И в этой главе мы о них подробно поговорим;
- в главе 6 речь пойдет о связи между архитектурой современного аппаратного обеспечения и реализацией кода на Python.

С учетом нюансов архитектуры «железа» наиболее производительным может оказаться код, который внешне не выглядит эффективным. К примеру, иногда работа со сжатыми данными может оказаться более эффективной в сравнении с несжатыми даже с учетом накладных расходов, связанных с их распаковкой. В этой главе мы также поговорим о том, как на реализацию кода на Python влияют архитектура центрального процессора, память, хранилище и сеть. Попутно мы познакомимся с библиотекой NumExpr, позволяющей повысить эффективность кода NumPy за счет использования характерных свойств архитектуры аппаратного обеспечения.

Часть III (главы 7–8). Приложения и библиотеки для современной обработки данных. Как ясно из названия, в этой части книги мы поговорим о распространенных приложениях и библиотеках, позволяющих оптимизировать процесс обработки данных:

- в главе 7 поработаем с библиотекой pandas, позволяющей максимально эффективно оперировать с табличными данными в виде датафреймов в Python. Мы рассмотрим различные техники оптимизации кода, связанные с этой библиотекой. В отличие от большинства глав этой книги в этой главе мы будем возвращаться к тому, что уже упоминалось ранее. Библиотека pandas работает поверх NumPy, так что вспомним кое-что из главы 4 и разработаем методы оптимизации pandas, связанные с пакетом NumPy. Кроме того, мы посмотрим, как можно повысить эффективность pandas с применением библиотеки NumExpr и расширения Cython. Наконец, познакомимся с Arrow – библиотекой, которая, помимо остального функционала, может быть использована для повышения эффективности обработки датафреймов в pandas;
- глава 8 будет целиком посвящена вопросам хранения данных. Мы рассмотрим библиотеку Parquet, позволяющую эффективно обрабатывать колоночные данные, и Zarr, служащую для обработки очень больших данных в виде массивов на диске. Мы также начнем разговор о том, как справляться с наборами данных, объем которых превышает доступные ресурсы памяти.

Часть IV (главы 9–10). Продвинутые возможности. В заключительной части книги мы поговорим о двух не связанных друг с другом темах: работе с графическим процессором (GPU) и использовании библиотеки Dask:

- в главе 9 научимся использовать ресурсы GPU при обработке больших данных. Мы обратим внимание на то, что архитектура графического процессора, предполагающая наличие большого количества простых ядер, хорошо подходит для решения актуальных задач из области науки о данных. Мы опробуем два различных подхода к использованию ресурсов GPU. Сначала поговорим о существующих библиотеках с возможно-

стями, схожими с уже известными вам, таких как CuPy – версии библиотеки NumPy для работы с графическим процессором. А затем узнаем, как писать код на Python, который будет выполняться с использованием ресурсов GPU;

- в главе 10 мы познакомимся с библиотекой Dask, позволяющей писать параллельный масштабируемый код с использованием ресурсов множества машин, как локальных, так и облачных. При всей своей кажущейся сложности эта библиотека предлагает интерфейс, очень похожий на уже знакомые вам библиотеки NumPy и pandas.

Книга также содержит два приложения:

- в приложении А содержатся инструкции для установки и настройки программного обеспечения, необходимого для проверки примеров из этой книги;
- в приложении Б обсуждается компилятор Numba, являющийся альтернативой расширению Cython в области генерирования эффективного низкоуровневого кода. Cython и Numba – это два основных инструмента для создания производительного кода на Python. Для решения насущных задач лично я рекомендую использовать Numba. Почему же я посвятил Cython целую главу, а компилятор Numba оставил для приложения? Причина в том, что главной целью этой книги является помощь в написании эффективного кода в рамках экосистемы Python, а расширение Cython, несмотря на дополнительные сложности, позволяет копнуть глубже в понимании того, что происходит на самом деле.

О сопроводительном коде

В данной книге содержится масса примеров исходного кода как в отдельных листингах, так и внутри обычного текста. В обоих случаях исходный код будет написан `моноширинным шрифтом` для его выделения на фоне описательного текста.

Стоит отметить, что в большинстве случаев мы вынуждены были переформатировать исходный код, добавив переносы строк и отступы для лучшей читаемости на страницах книги. Также мы удалили из кода некоторые комментарии, которые дублируются рядом с помощью аннотаций. Таких специальных аннотаций к коду будет достаточно много – с помощью них мы постарались выделить особо важные моменты в листингах.

Вы можете загрузить исполняемые примеры кода из онлайн-версии книги, находящейся по адресу https://livebook.manning.com/book/fast-python. В полном виде исходные коды собраны на GitHub по адресу https://github.com/tiagoantao/python-performance, а также на сайте издательств https://www.manning.com и https://dmkpress.com. При обнаружении ошибок или изменении требований исполь-

зованных библиотек мы будем обновлять содержимое исходных кодов. Таким образом, в текущем виде код в репозитории может отличаться от присутствующего в книге. Сопроводительные материалы в репозитории организованы по главам.

В целом приведенный в книге код был серьезно адаптирован для нужд печати. К примеру, на практике я являюсь ярым сторонником длинных и понятных имен переменных, но в связи с ограничениями книги такие имена здесь не подходят. Я сделал все, чтобы сохранить выразительность имен переменных и соблюсти все требования стандартов в Python, таких как PEP8, но читаемость кода в книге была поставлена мной во главу угла. То же самое верно и для аннотаций типов: я бы с удовольствием их использовал, но они нарушают читаемость кода.

В большинстве случаев код, приведенный в этой книге, будет работать со стандартным интерпретатором Python. Иногда для выполнения кода потребуется IPython, особенно когда речь пойдет об анализе производительности. Также вы можете использовать Jupyter Notebook.

Инструкции по установке и настройке необходимого программного обеспечения можно найти в приложении А. Если в какой-то главе или разделе потребуется установить дополнительные программы или библиотеки, об этом будет упомянуто отдельно.

Программное и аппаратное обеспечение

Исходный код, приведенный в книге, вы можете запускать на любой операционной системе. В то же время в большинстве случаев рабочие проекты развертываются в среде Linux, так что эта операционная система может считаться предпочтительной. На MacOS X тоже никаких проблем с адаптацией не возникнет. Что касается Windows, я рекомендую установить подсистему Linux для Windows (WSL – Windows Subsystem for Linux).

Альтернативой операционным системам может считаться запуск кода в Docker. Вы можете использовать образы Docker, содержащиеся в репозитории. С помощью Docker вы можете создать окружение Linux в виде контейнера для запуска приведенного в книге кода.

В качестве минимальных требований для запуска кода я бы порекомендовал конфигурацию с 16 Гб памяти и 150 Гб свободного дискового пространства. В главе 9 мы будем обсуждать темы, связанные с использованием вычислительных ресурсов графического процессора, и для проверки кода вам потребуется GPU от NVIDIA с микроархитектурой Pascal. Большинство GPU, выпущенных компанией за последние пять лет, будут отвечать этому требованию. Все остальные инструкции по установке и настройке программного и аппаратного обеспечения можно найти в приложении А.

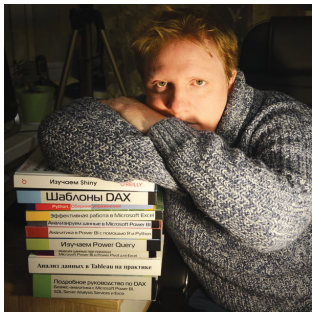
Об авторе

Тиаго Антао (Tiago Rodrigues Antão) обладает степенью бакалавра технических наук в области информатики и докторской степенью в области биоинформатики. В настоящее время работает в сфере биотехнологии. В своей работе Тиаго на постоянной основе использует язык Python и все его популярные библиотеки для выполнения научных расчетов и анализа данных. Для оптимизации критических алгоритмов использует низкоуровневые языки программирования C и Rust. В настоящее время Тиаго занимается разработкой инфраструктуры на базе Amazon AWS, но на протяжении большей части карьеры использовал локальные вычислительные кластеры.

Помимо плодотворной работы в индустрии, Тиаго прошел две постдокторантуры по анализу данных в Кембриджском и Оксфордском университетах. В качестве ученого-исследователя при университете Монтаны он с нуля создал научно-вычислительную инфраструктуру для анализа биологических данных.

Тиаго является одним из создателей популярного набора модулей *Biopython*, написанных на Python, а также автором книги *Bioinformatics with Python Cookbook*, увидевшей свет в 2022 году уже в третьем издании. Кроме того, Тиаго написал большое количество важных научных докладов и статей по биоинформатике.

О переводчике



Александр Гинько, обладающий богатым опытом работы в сфере ИТ и более десяти лет посвятивший переводам книг и статей на самые разные темы, в последние годы специализируется на переводе книг в области бизнес-аналитики и программирования для издательства «ДМК Пресс» по направлениям Python, SQL, Power BI, DAX, Excel, Power Query, Tableau, R... На данный момент в активе Александра уже более 20 книг, включая одну авторскую, и он продолжает плодотворно работать над переводом новых.

Помимо перевода книг, Александр ведет свой канал в Telegram (https://t.me/alexanderginko_books), на котором вы можете из первых уст получить ответы на все интересующие вас вопросы об уже переведенных книгах, находящихся в работе и запланированных на будущее. Также на канале можно найти промокоды на все книги Александра для покупки книг на сайте издательства «ДМК Пресс» с большими скидками.

Часть I

Фундаментальные подходы

В первой части книги мы поговорим об основных подходах и предпосылках в отношении эффективности кода, написанного на языке Python. Мы пройдемся по популярным библиотекам языка и основным структурам данных, а также посмотрим, как в Python (без использования внешних пакетов) реализованы техники параллельных вычислений. Отдельная глава будет посвящена оптимизации вычислений с использованием библиотеки NumPy. И хотя теоретически библиотека NumPy является внешней, ее роль в современных технологиях обработки данных столь велика, что мы включили ее в эту часть книги наряду с другими фундаментальными подходами к вычислениям.

1 Острая нехватка производительности при обработке данных

В этой главе мы обсудим следующие темы:

- способы борьбы с экспоненциальным ростом объемов данных;
- сравнение традиционных и современных вычислительных архитектур;
- роль и недостатки языка Python в современном анализе данных;
- техники для реализации эффективных вычислительных решений на Python.

В настоящее время данные собираются в невероятном количестве, на огромных скоростях и из самых разнообразных источников. Они собираются даже безотносительно к тому, будут использоваться в ближайшем будущем или нет, есть ли место для их хранения, время и мощности для обработки, анализа и изучения. Еще до того, как аналитики решат, как использовать эти данные, а разработчики и управленцы поймут, как на их основе создавать новые служ-

бы и продукты, инженеры-программисты должны найти способы для их хранения и обработки. И сейчас больше, чем когда-либо, им необходимо пытаться использовать любые возможности для повышения эффективности процесса сбора данных и оптимизации их хранения.

В этой книге я попытался собрать все известные мне стратегии по оптимизации обработки и хранения данных – лично я всеми ими пользуюсь в своей ежедневной работе. Просто выделить больше машин для обработки информации зачастую бывает невозможно, да это и не помогает. Таким образом, мы будем больше говорить об эффективных средствах, имеющихся в нашем распоряжении, как то: оптимизация кода, адаптация под архитектуру программного и аппаратного обеспечения и, конечно, нюансы языка, библиотек и экосистемы Python в целом.

Python стал наиболее предпочтительным языком или некой связующей прослойкой для выполнения всей рутинной работы в условиях этого нескончаемого потока или даже потопа данных. Популярность Python в науке о данных и инженерии данных обусловила рост его востребованности во всех остальных сферах, что сделало этот язык, по данным различных исследований, одним из трех наиболее популярных языков программирования. Язык Python имеет целый ряд преимуществ, но без недостатков применительно к обработке больших данных в нем не обошлось. В частности, это касается вопросов скорости обработки данных. К счастью, существует множество подходов для устранения этих недостатков, которые способны значительно повысить эффективность языка Python при обработке больших данных.

Но, прежде чем решать проблемы, необходимо точно и четко их проговорить, чем мы по большей части и будем заниматься в данной главе. Мы поговорим о последствиях наводнения всех сфер нашей жизни данными и обрисуем проблемы, с которыми нам, как инженерам, приходится сталкиваться при обработке этого бесконечного потока. После этого обсудим роль аппаратного обеспечения, сети и архитектуры облачных ресурсов, чтобы понять, что прежние подходы с увеличением рабочей частоты центрального процессора больше не работают. Затем мы проговорим проблемы, с которыми сталкивается язык Python при обработке больших объемов данных, включая управление потоками и глобальную блокировку интерпретатора, применяемую в CPython. И только после осознания того, что для повышения эффективности кода на Python нам необходимы новые пути, мы представим решения, которые будем реализовывать на протяжении всей книги.

1.1. Насколько велик всемирный потоп данных?

Возможно, вы слышали о существовании двух законов, носящих имена Мура (Moore) и Эдхольма (Edholm), которые в совокупности рисуют очень драматичную картину, связанную с экспоненциальным ростом собираемых данных и отставанием индустрии по обработке этих самых данных. Так, *закон Эдхольма* гласит, что объем собираемых данных с помощью средств телекоммуникации удваивается каждые 18 месяцев. В то же время *закон Мура* утверждает, что количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 24 месяца. С целью упрощения мы можем применить закон Эдхольма ко всем собираемым данным, а закон Мура интерпретировать как индикатор доступных ресурсов вычислительного оборудования. Если совместить два этих закона, мы получим запаздывание развития технологий относительно роста объема данных для обработки и хранения на полгода. Поскольку экспоненциальный рост трудно описать словами, лучше будет взглянуть на график, показанный на рис. 1.1.

Ситуацию на этом графике можно описать как борьбу между тем, что нам нужно проанализировать (закон Эдхольма), и тем, с помощью чего мы собираемся проводить анализ (закон Мура). При этом на графике перспектива показана даже в более оптимистичном свете по сравнению с реальностью. Почему? Узнаем в главе 6, когда будем рассматривать закон Мура применительно к современной архитектуре центральных процессоров.

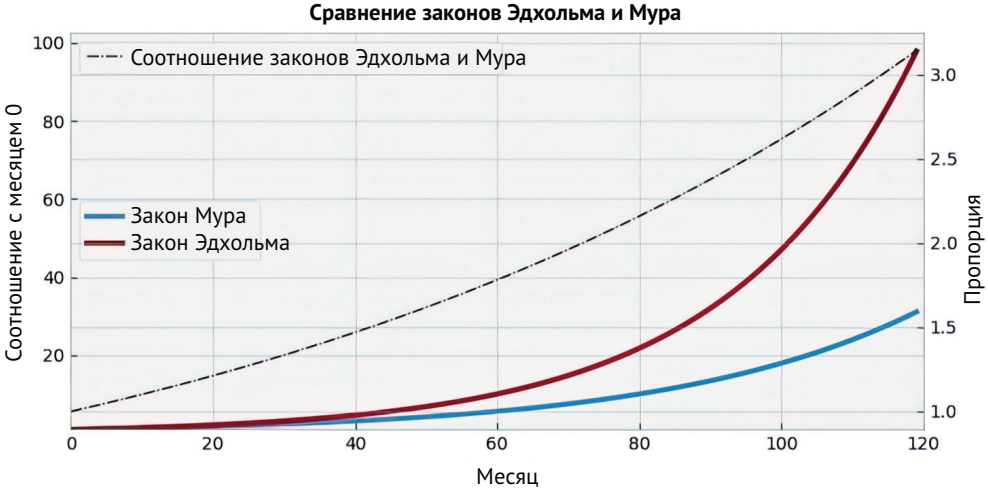


Рис. 1.1. Соотношение законов Эдхольма и Мура предвещает постоянно увеличивающееся отставание вычислительных возможностей от доступного объема данных