МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

ИНСТРУМЕНТЫ РАСПАРАЛЛЕЛИВАНИЯ В ЯЗЫКЕ PYTHON

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Королевой Татьяны Сергеевны

Научный руководитель
доцент, к. ф.-м. н. _____ А. Д. Панферов

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ 5
ЗАКЛЮЧЕНИЕ 9
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 10

ВВЕДЕНИЕ

Оценка популярности того или иного языка программирования сложная и многопараметрическая задача. Тем не менее одним из лидеров, безусловно, в настоящее время является Python [1]. И его популярность растет на протяжении нескольких последних лет. Можно уверенно сказать, что этим он обязан использованию в приложениях по обработке и анализу данных. Хотя его востребованность в программировании для WEB остается устойчиво высокой, рост обеспечивает именно относительно новый сегмент работы с данными. В этом сегменте пользователи очень часто не являются профессиональными программистами и простота синтаксиса, гибкость и расширяемость за счет имеющихся библиотек под любые типы задач оказываются решающими.

Имеется определенное противоречие между проблематикой работы с большими объёмами данных, при которой очень существенна роль высокой производительности приложения, возможности его работы в параллельном режиме, и выбором этого языка, разработанного очень давно, когда параллелизм был прерлгативой уникальных высокопроизводительных систем. Хотя Python прошел длительный путь развития, первым и главным его недостатком эксперты называют низкую производительность. Это определяется тем, что данный язык интерпретируемый.

Пока развитие технологий обепечивало уверенный рост производительности ЭВМ, проблема недостаточного быстродействия решалась с появлением процессоров нового поколения. Однако на уровне достижения тактовых частот 2 - 4 ГГ их дальнейший рост стал невозможен по технологическим причинам. Для повышения производительности процессоров пришлось перейти к использованию в них аппаратного параллелизма [2] путем внедрения многоядерных решений. В современных настольных системах трудно найти процессоры с количеством ядер менее 4-х. Такой путь развития потребовал соответствующей адаптации программных средств.

Целью представляемой работы является изучение инструментов распараллеливания программ, имеющихся в распоряжении программистов, использующих Python и эффективности этих средств.

В первой представлен обзор принципов организации параллельных вычислителей и программ. Во второй главе излагаются результаты изучения

возможностей Python по организации мультипотокового и мультипроцессорного режима выполнения заданий.

В третьей главе представлен пример решения задачи по реализации мультапроцессного параллелизвма и результаты его тестирования.

1 КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первой главе: Параллелизм. Причины возникновения и особенности рассматриваются причины массового использования аппаратного параллелизма в современных вычислительных системах. Излагаются основные концепции организации вычислительного процесса в параллельном режиме. Особенности реализации алгоритмов при их распараллеливании [3].

Во второй главе предметом рассмотрения является язык Python и имеющиеся в нем инструменты для организации параллельных высчислений. Кратко была привелена история создания языка, этапы усовершенствования его версий, и что наиболее важно причины популярности этого языка программирования. Синтаксис Python выделяет его на фоне прочих языков программирования. Его сходства с английским языком позволяет читать код не профессиональному пользователю. Поэтому простота синтаксиса рассматривается в качестве главного достоинства этого языка, существенно снижая порог входа в сообщество его пользователей. Это стало одной из причин популярности Python в среде аналитиков различного направления, занимающихся сбором, обработкой и анализом данных [4, 5].

Однако быстродействие никогда не было в числе достоинств этого языка. Рост вычислительных возможностей ЭВМ на первом этапе его развития позволял решать все более и более сложные задачи, но с переходом к параллелизму на уровне процессоров и фактическим прекеращением роста производительности отдельных вычислительных ядер актуальной стала проблема использования параллелизма для роста производительности приложений, написанных на этом языке. В главе описываются соответствующие инструменты и методы их использования [6].

В третьей главе демонстрировалось распараллеливание с использованием процессов в Python

В качестве примера использования параллелизма рассмотрена следующая задача. В трёхмерном пространстве заданы N точек v0, v1, ..., vN. Требуется для каждой пары точек вычислить функцию, зависящую от расстояния между ними. Результат будет представлять собой матрицу NxN со значениями этой функции. В качестве функции использовалось: $f = r^3/12 + r^2/6$ [7].

В данной задаче каждая строка матрицы может вычисляться независи-

мо. С помощию программы на Python, использующей модуль multiprocessing, из каждых нескольких строк матрицы формировались независимые задания, распределявшиеся между процессами.

Тестирование проводилось на вычислительной системе с процессором Intel(R) Core(TM) i5 с базовой тактовой частотой $2.6GHz$ и объёмом ОЗУ $32GB$. Тест выполнялся в режиме последовательного решения задачи для заданного значения числа точек N с использованием одного потока и затем в мультипотоковом режиме при различном числе потоков. Для первой серии тестов использовался диапазон значений $19900 \leq N \leq 20090$ с шагом 10. На каждый процесс выделялось 100 заданий. Результат сравнения выводился в виде графика, показывавшего время выполнения заданий (в секундах) одним процессом - синяя линия и заданным количеством процессов в мультипроцессном режиме - оранжевая линия (Рисунок 1).

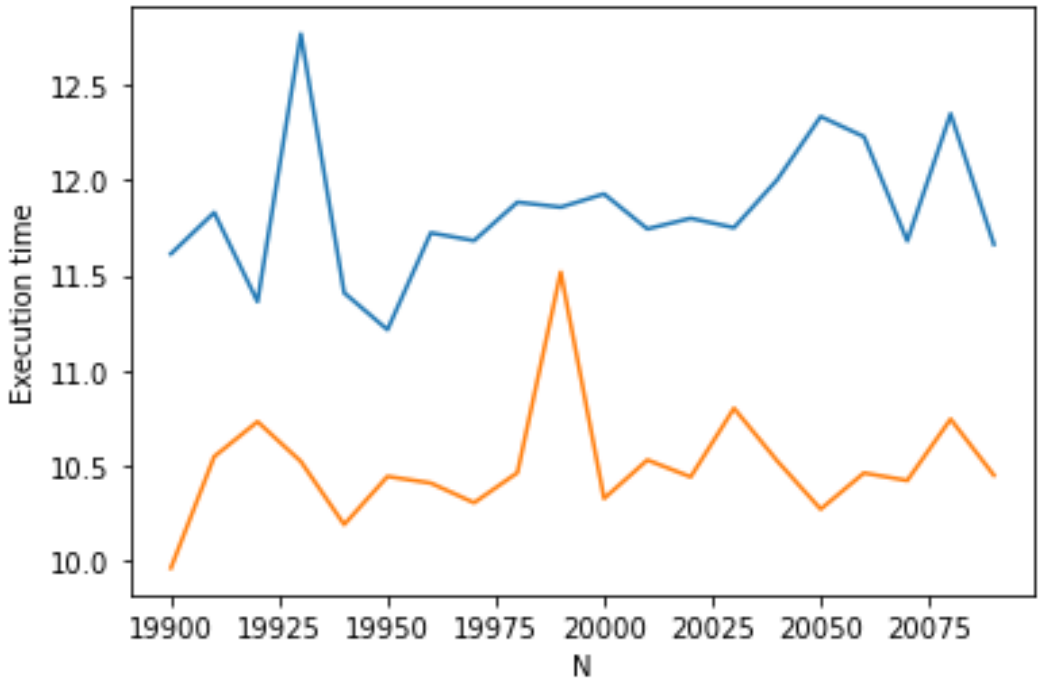

Рисунок 1 – Сравнение времени решения тестовой задачи в однопроцессном и двухпроцессном режиме

Аналогично было проведено тестирование для большего количества процессов. На Рисуноке 2 приведены сравнительные результаты для четырех и пяти процессов. Сравнивая Рисунки 1 и 2 можно сделать вывод, что переход от двуз к четырем процессам обеспечил небольшое дополнительное ускорение. Дальнейшее увеличение числа процессов не приводит к значимому эффекту.

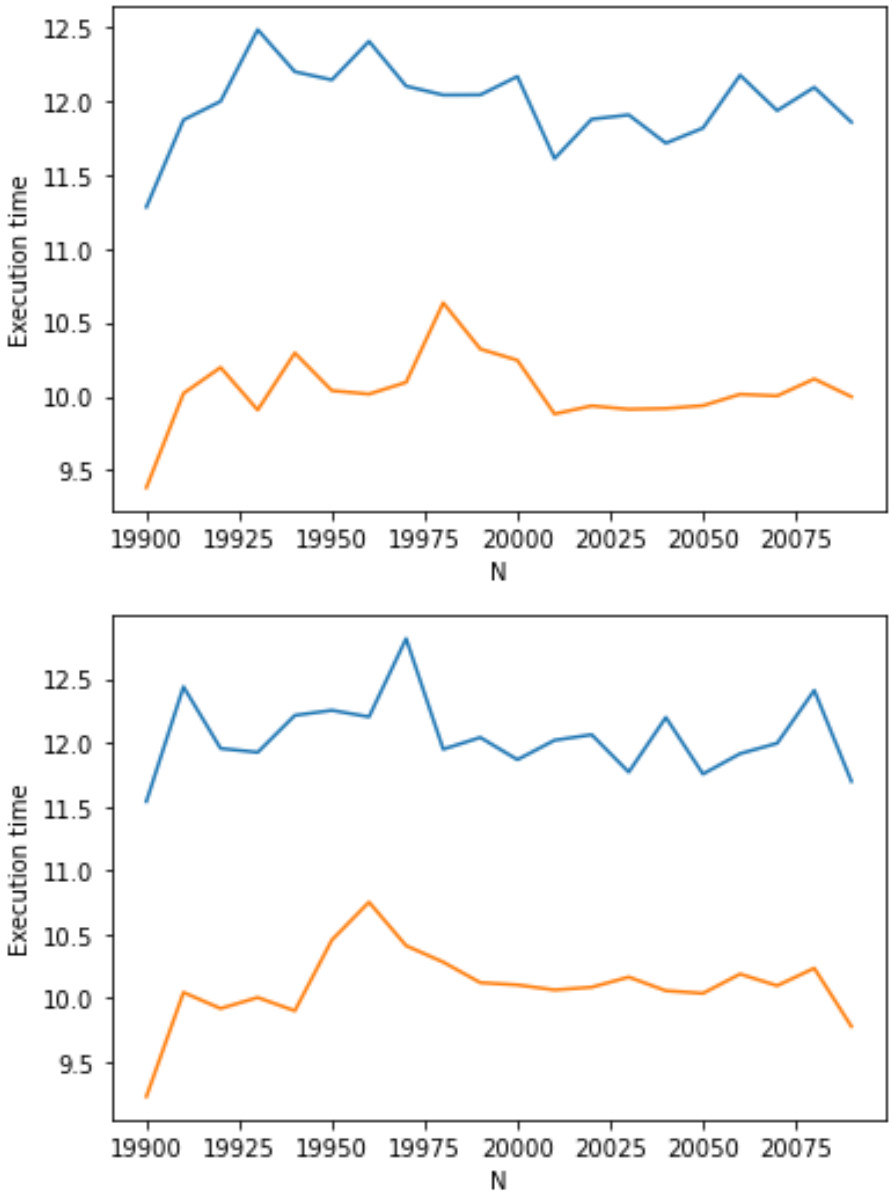

Рисунок 2 – Времена решения тестовых задач при использовании четырех и пяти процессов (в сравнении с одним процессом)

Процесс вычисления контролировался с использованием системного монитора ресурсов. На Рисуноке 3 представлен график загрузки процессора (6 физических ядер) при чередующемся выполнении тестового задания в однопроцессном режиме и в четырехпроцессном режиме. Отчетливо прослеживаются переходы между режимами работы программы. Это позволяет сделать вывод, что в данном случае деествительно дополнительные процессы задействуют новые аппаратные ресурсы.

На Рисуноке 4 представлены результаты еще одного теста. В этом случае использовался диапазон значений $23900 \leq N \leq 24090$. Шаг и количество заданий на один процесс прежнее. Сравниваются времена выполнения за-

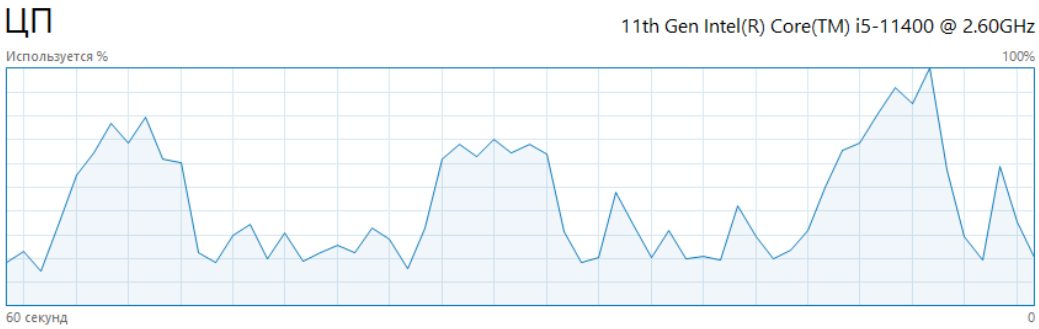

Рисунок 3 – Чередование режима работы один процесс / четаре процесса

даний одним процессом и двумя прпоцессами. В данном случае результаты

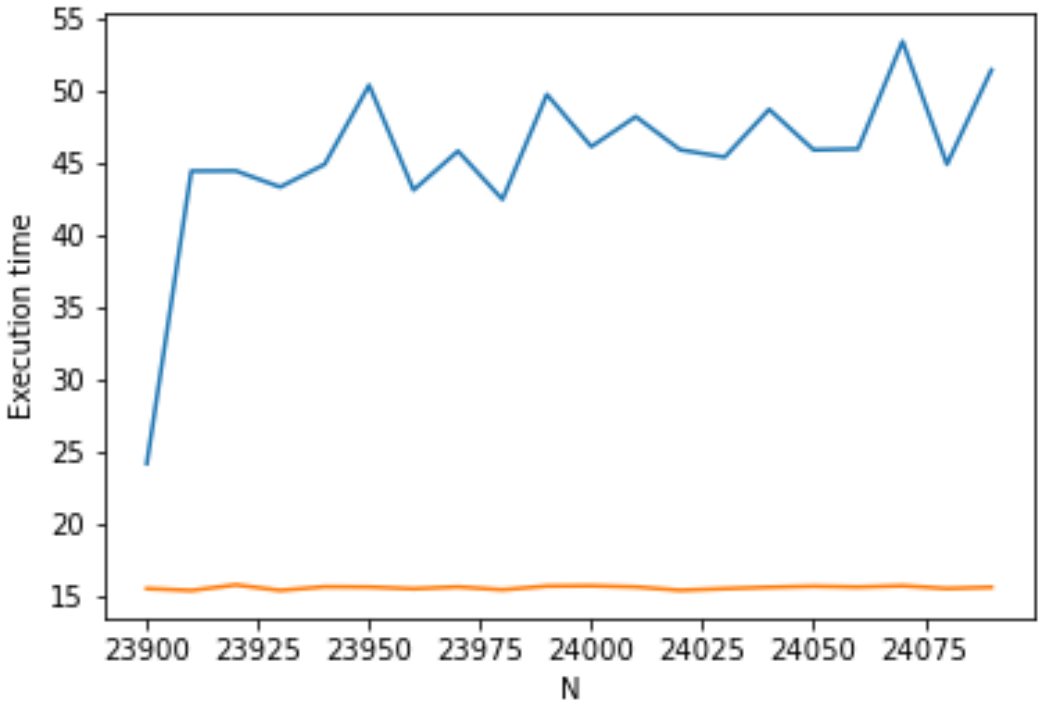

Рисунок 4 – Сравнение времени решения тестовой задачи в однопроцессном и двухпроцессном режиме при большем объёме обрабатываемых данных

работы в режиме использования двух процессов очень стабильны а ускорение, продемонстрированное на всех точках кроме первой, даже более чем двухкратно. Можно предположить, что последнее обстоятельство связано с особенностями рабрты с памятью при больших объёмах данных в заданиях.

Таким образом из представленных результатов можно сделать вывод, что использование параллелизма, в частности использование пакета multiprocessing языка программирования Python позволяет ускорить работу высоконагруженных приложений, предназначенных для большого объёма вычислений.

ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены принципы и способы использования параллелизма для ускорения выполнения различных вычислительных задач с применением соответствующих аппаратных ресурсов. Были изучены инструменты распараллеливания, присутствующие в современной версии языка программирования Python. Проанализированы их возможности и ограничения, преимущества и недостатки. Для демонстрации полученных оценок представлено тестовое приложение, позволяющее выполнять сравнение скорости выполнения высоконагруженных приложений в однопроцессном и мультипроцессных вариантах. Результаты выполненных тестов демонстрируют реальность выигрыша в производительности при использовании мультипроцессорных архитектур.

Поставленная цель была успешно достигнута, задачи решены в полном объёме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Баденко В.Л. Высокопроизводительные вычисления. Учебное пособие. — СПб.: Изд-во Политехн. ун-та, 2010. — 180 С.

2 Барский А.Б. Параллельные информационные технологии: Учебное пособие/А.Б. Барский.-М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007.-503 с.: ил.,таб.-(серия «Основы информационных технологий»)- с.20-28, с.56-58.

3 Закон Мура [Электронный ресурс] – URL:https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%9C%D1%83%D1%80%D0%B0 (дата обращения 20.05.22) – Загл. с экрана– Яз. рус.

4 ГОСТ 19781-90: Обеспечение систем обработки информации программное. Термины и определения

5 Параллельные вычисления: учеб. пособие /Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 149 С.

6 Валях Е. Последовательно-параллельные вычисления. М.: Мир. 1985. 456 С.

7 Параллельное программирование в Python при помощи multiprocessing и shared array: [Электронный ресурс] – URL:https://habr.com/ru/post/167503/ (дата обращения 9.04.22) – Загл. с экрана– Яз. англ.