Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИИВЕРСИТЕТ ИМЕНИ Н.Г.
ЧЕРНЫШЕВСКОГО»

Кафедра физики открытых систем

Моделирование системы фазовых осцилляторов Курамото на языке программирования Python с использованием технологий параллельных вычислений

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 431 группы
направления 09.03.02 «Информационные системы и технологии»
Факультета нелинейных процессов
Десятова Игоря Владимировича

Научный руководитель
профессор, д.ф.-м.н., доцент _____ О.И. Москаленко
подпись, дата

Зав. Кафедрой физики открытых систем
д.ф.-м.н., профессор _____ А.А. Короновский
подпись, дата

Саратов 2018

ВВЕДЕНИЕ

В последние годы очевидным становится доминация многоядерных вычислительных платформ на компьютерном рынке. Всё чаще можно видеть, как ранее написанные программы становятся невостребованными на новых процессорах и видеокартах, т.к. не были рассчитаны на произведение операций в параллельном режиме. В данном свете программирование с использованием параллельных вычислений приобретает высокую актуальность.

Другим аспектом является моделирование сложных систем с большим числом параметров, применяемое как для научных вычислений, так и для мультимедиа. Модель фазовых осцилляторов Курамото является примером такой системы.

Целью бакалаврской работы является написание программы, моделирующей систему фазовых осцилляторов Курамото, на языке программирования Python с использованием технологий параллельных вычислений на теоретической основе ресурсов, и оценка её производительности в зависимости от входных параметров.

Задачи:

- Реализация метода распараллеливания на базе технологии MPI
- Реализация метода распараллеливания на базе технологии OpenCL
- Поиск оптимальных условий исследуемой модели фазовых осцилляторов Курамото

Объектом исследования является сеть фазовых осцилляторов Курамото.

Предметом исследования является использование технологий MPI и OpenCL.

Данная работа состоит из введения, трёх глав, заключения и списка литературы.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Основная состоит из трёх глав, отвечающих за описание «программных приложений», «моделирования с помощью технологий MPI и OpenCL» и «оценки производительности программы».

1 Программные приложения

1.1 Параллельные вычисления

За последние годы высокая производительность программ стала важным факторов для многих исследователей. Слияние качественного программного обеспечения с открытым исходным кодом и товарного оборудования обеспечило широкую популярность кластеров Beowulf и кластеров рабочих станций.

Из многих моделей выполнения параллельных вычислений, модель с передачей сообщений оказалась наиболее эффективной. Эта парадигма особенно подходит для архитектуры распределенной памяти и используется на сегодняшний день в самых требовательных научных и технических приложениях, связанных с моделированием, имитацией, проектированием и обработкой сигналов. Однако написание программ под разные виды ЭВМ на основе передачи сообщений в прошлом было нетривиальной задачей из-за множества несовместимых компонентов, с которыми сталкивались разработчики. Положение вещей изменилось в лучшую сторону после того, как MPI Forum опубликовал стандартную спецификацию.

Высокопроизводительные вычисления традиционно связаны с разработкой программного обеспечения с использованием компилируемых языков. Однако в типичных прикладных программах только небольшая часть кода достаточно критична по времени, чтобы требовать эффективности компилируемых языков. Остальная часть кода, как правило, связана с управлением памятью, обработкой ошибок, вводом/выводом и взаимодействием с пользователем, и обычно это наиболее подверженные ошибкам и объёмные участки кода для написания и отладки во всем процессе разработки. Интерпретируемые языки высокого уровня могут быть действительно полезными для таких задач.

В настоящей бакалаврской работе используется «MPI for Python» - хорошо известный свободно распространяемый программный пакет для языка

Python, позволяющий приложениям использовать несколько процессоров с использованием стандартного MPI-интерфейса.

1.2 Python

Python - это современный, простой в освоении, мощный язык программирования. Он имеет эффективные структуры данных высокого уровня и простой, но эффективный подход к объектно-ориентированному программированию с динамической типизацией и динамической привязкой. Он поддерживает модули и пакеты, что способствует модульности программы и повторному использованию кода. Минималистичный синтаксис Python вместе с его интерпретируемой природой делает его идеальным языком для сценариев и быстрой разработки приложений во многих областях на большинстве платформ.

Интерпретатор Python и большая стандартная библиотека доступны бесплатно для всех основных платформ и могут свободно распространяться. Он легко дополняется новыми функциями и типами данных, реализованными на C или C++. Python также подходит как язык расширений для настраиваемых приложений.

Python является идеальным кандидатом для написания частей более высокого уровня широкомасштабных научных приложений и управления моделированием в параллельных архитектурах, таких как ПК кластеры или SMP. Библиотеки Python быстро развиваются, легко сохраняются и могут достигать высокой степени интеграции с другими библиотеками, написанными на компилируемых языках.

Также Python хорошо подходит для написания небольших программ, разработки тестов и т.д.

По сравнению с такими аналогами как Perl и Ruby, Python имеет ряд преимуществ:

- Единообразный, строгий синтаксис.

- Предельно низкий порог вхождения

- Огромное сообщество пользователей Python, а, следовательно, большое количество книг и ответов на проблемы

- Большая коллекция готовых библиотек

- Строгая динамическая типизация

- Сборщик неиспользуемых ссылок на данные, включая циклические ссылки

- Возможность передавать не примитивные типы и объекты по ссылке

1.3 Модель Курамото

В качестве модели, вычисление которой будут распараллелено, выступает модель Курамото – одна из наиболее представительных моделей связанных фазовых осцилляторов в которой могут наблюдаться режимы синхронизации.

Модель фазовых осцилляторов Курамото описывается системой дифференциальных уравнений вида:

$$\frac{d\theta_i}{dt} = \omega_i + \lambda \sum_{j=1}^{N} A_{ij} \sin\left(\theta_j - \theta_i\right),\ i = 1, \dots, N, \qquad (1)$$

i, j – номер осциллятора;

λ - коэффициент связи;

ω_i - собственная частота i-го осциллятора;

θ_i – фаза колебаний i-го осциллятора;

A – матрица связи;

N – число осцилляторов в системе.

Параметр порядка r рассчитывается по формуле:

$$re^{i\psi} = \frac{1}{N}\sum_{j=1}^{N} e^{i\theta_j}, \qquad (2)$$

Здесь $0 \leq r(t) \leq 1$ измеряет когерентность колебаний совокупности осцилляторов, а $\psi(t)$ - среднюю фазу.

1.4 Технология MPI

MPI- это стандартизованная и портативная система передачи сообщений, предназначенная для работы на широком спектре параллельных компьютеров. Стандарт определяет синтаксис и семантику библиотечных подпрограмм и позволяет пользователям писать переносные программы на основных языках научного программирования.

Открытый проект Open MPI - это реализация интерфейса с открытым исходным кодом, разработанная и поддерживаемая консорциумом академических, исследовательских и отраслевых партнеров. Таким образом, Open MPI способен объединить знания, технологии и ресурсы со всех сторон сообщества высокопроизводительных вычислений, чтобы создать лучшую доступную библиотеку MPI. Технология Open MPI предлагает преимущества для поставщиков систем и программного обеспечения, разработчиков приложений и исследователей в области компьютерных наук.

1.5 Интеграция MPI и Python

Взаимодействия Python с интерфейсом MPI происходят через API - интерфейс прикладного программирования, содержащийся в библиотеке «mpi4py», известной как «MPI for Python».

MPI - message passing interface - библиотека функций, предназначенная для поддержки работы параллельных процессов в терминах передачи сообщений.

Все процессы содержатся в группе с идентификатором *MPI_COMM_WORLD*.

Номер процесса (ранг) - целое неотрицательное число. Является уникальным атрибутом каждого процесса.

Процессы управляются посредством коллективной коммуникации.

Особенностью коллективной коммуникации является то, что она подразумевает точку синхронизации между процессами. Это означает, что все процессы должны достигнуть точки в своем коде, прежде чем они смогут снова начать выполнение.

1.6 Технология OpenCL

OpenCL (Open Computing Language) - это открытый стандарт для кросс-платформенного параллельного программирования различных процессоров, имеющихся на персональных компьютерах, серверах, мобильных устройствах и встроенных платформах. OpenCL значительно улучшает скорость широкого спектра приложений во многих рыночных категориях, включая игровые и развлекательные области, научное и медицинское программное обеспечение, профессиональные креативные инструменты, обучение нейронных сетей.

Плюсами данной платформы является высокий уровень используемых операций, недостатками – закрытость технологии и поддержка исключительно видеокарт NVIDIA, а значит низкая переносимость на другие платформы.

2. Моделирование с помощью технологий MPI и OpenCL.

2.1. Модели распараллеливания вычислений.

Для распараллеливания вычислений как для MPI, так и для OpenCL было принято решение использовать модель *Единой программы, множественных данных SPMD*. Ограничение одной программой означает, что все задачи выполняют свою копию одной и той же программы одновременно. Эта программа может осуществлять операции с потоками, передачей сообщений, с

данными параллельными или гибридными. Наличие множественных данных показывает, что для всех задач могут быть использованы разные данные.

2.2 Распараллеливание вычислений MPI

Согласно модели SPMD, дан начальный вектор фаз *initial_conditions*. Каждая фаза в векторе фаз соответствует осциллятору под тем же номером. Каждую итерацию вычисляется последующий вектор фаз такой же длины, исходя из данных предыдущего.

Для оптимальной обработки данных необходима равномерная нагрузка на все узлы MPI. Для этого поделим количество осцилляторов osc на количество узлов size, а остаток от деления распределим по одному осциллятору на узел.

2.3 Распараллеливание вычислений OpenCL

Модель представления данных аналогична модели в пункте 2.2, но схема разбиения данных отличается. Данные разбиваются по группам максимально возможной малости – здесь каждая фаза будет является самостоятельным блоком данных. Задача по распределению уже разбитых на группы данных по вычислительным модулям ложиться на программное обеспечение видеокарты, опирающееся в принципах работы на стандарт OpenCL.

Если число осцилляторов больше числа физических процессоров видеокарты, то OpenCL самостоятельно распределит нагрузку между всеми доступными процессорами.

2.4 Результаты моделирования системы осцилляторов

Вычислив фазы осцилляторов по указанной в теории формуле, получим значения фаз в радианах. Для более наглядного представления преобразуем фазы осцилляторов из радиан в безразмерную величину по формуле:

$$\theta_i = \sin(\theta_{rad_i}\ mod\ 2\pi), i = 1, 2, .., osc. \quad (3)$$

Где θ_{rad} – фаза i-го осциллятора, θ – sin фазы i-го осциллятора, i – номер осциллятора, *osc* – число осцилляторов в системе.

2.5 Проверка модели на корректность по результатам её работы.

Рассматривается зависимость параметра r от λ. Кривая не является гладкой т.к. для каждой точки при фиксированном параметре λ берётся усреднённое значение параметра r, рассчитываемое по формуле:

$$r(\lambda) =< r(t,\ \lambda) >_t, t \in [50, 100] \quad (4)$$

2.6 Подсчёт времени выполнения программы с MPI

Измеряется время выполнения программы в зависимости от количества осцилляторов. Количество итераций при моделировании одной системы iter=200. Время выполнения снижается при увеличении количества используемых узлов.

2.7 Подсчёт времени выполнения программы с OpenCL

Измерим зависимость времени выполнения программы от количества осцилляторов в системе. При увеличении числа осцилляторов в n раз количество вычислений увеличивается n^2 раз, т.к. все осцилляторы в моделируемой системе связаны друг с другом по фазе, и добавление в полносвязную систему дополнительного элемента ведёт к квадратичному увеличению числа связей.

Однако, время выполнения растёт линейно до того, как количество физических процессоров становится меньше количества осцилляторов в системе для используемой в эксперименте видеокарты. Это напрямую связано с распараллеливанием модели, и тем, что, до этого свободный рабочий юнит берёт на себя работу по вычислению фазы для n+1 осциллятора, в итоге увеличение вычислительной нагрузки ограничивается добавлением в каждое уравнение системы дополнительного слагаемого, отвечающего за связь оставшихся осцилляторов с новым. Для всех осцилляторов связь с собственной фазой задаём равной нулю.

3 Оценка производительности программы

3.1 Общие понятия. Методы оценки производительности программы в зависимости от внешних и внутренних факторов

Для решения указанной задачи используем аппарат, разработанный в теории математического планирования эксперимента (МПЭ). Одна из основных идей планирования эксперимента состоит в использовании для исследуемого объекта кибернетической абстракции черного ящика.

Такая абстракция предполагает отказ от рассмотрения внутренних механизмов исследуемого явления или объекта из-за большой сложности. Анализ явления сводится к анализу входящих параметров, воздействующих на объект (факторов) и выходных характеристик (откликов).

3.2 Оценка производительности полученной программы

3.2.1 Факторы, влияющие на производительность программы

Фактор – это параметр, прямым или косвенным образом влияющий на производительность системы. Факторы тестируемой системы приведены ниже.

Таблица 1. Факторы тестируемой системы

Фактор	Описание	Минимальное значение	Максимальное значение
Iter	Количество итераций при моделировании одной системы	200	2000
Osc	количество осцилляторов в системе	10	1000
OpenCL или MPI	технология распараллеливания вычислений	-1 - MPI;	1 - OpenCL
N	количество вычислительных юнитов (узлов)	1	8

Параметр n для всех экспериментов берётся максимальным.

Сравниваются производительность вычислений с MPI на процессоре, использующем 8 вычислительных юнитов, и производительность вычислений с участием стандарта OpenCL на видеокарте Radeon 550 Series, содержащей в себе тоже 8 юнитов, размер рабочей группы каждого - 256.

3.2.2 Итог тестов. Время выполнения при различном сочетании факторов

Составим матрицу планирования. Согласно теории планирования эксперимента, в матрице планирования, представленной в таблице 2, будем использовать принятые обозначения: +1 соответствует верхнему уровню фактора; -1 соответствует нижнему уровню фактора.

Таблица 2. Матрица планирования

№	MPI/CL	iter	osc	Время (с)
1	-1	-1	-1	0.02908
2	-1	-1	1	22.56792
3	-1	1	-1	0.16434
4	-1	1	1	227.47706
5	1	-1	-1	0.07892
6	1	-1	1	0.17324
7	1	1	-1	0.47991
8	1	1	1	1.30784

3.2.3 Анализ результатов

Планируя эксперимент, целью являлось получить линейную модель. В случае нелинейности модели можно количественно оценить нелинейность, пользуясь полным факторным экспериментом, где учитывается зависимость эффекта одного фактора от уровня, на котором находится другой фактор.

Из полученных результатов раздела 3.2.3 можно утверждать, что нагрузка на ЭВМ увеличивается больше при увеличении количества осцилляторов, чем при увеличении количества итераций в указанных границах изменений. Также, таблица 2 показывает, что на используемой ЭВМ расчёты с использованием технологии OpenCL становятся производительнее на порядки по сравнению с аналогичными вычислениями с использованием MPI при увеличении числа осцилляторов в системе. Это можно объяснить большим количеством потоков,

которые может поддерживать видеокарта, а следственно большим числом одновременно обрабатываемых данных по сравнению с CPU.

ЗАКЛЮЧЕНИЕ

На основании проделанной работы, можно подвести следующие итоги:

1. Проведён обзор использованных в работе ПО и технологий параллельных вычислений.

2. Представлено описание распараллеливания производимых вычислений

3. Приведён анализ производительности полученного программного продукта

Рассмотрена производительность программы, которая использует технологию MPI, и производительность программы, которая использует технологию OpenCL, на оборудовании с сопоставимыми мощностями.

Распараллеливание вычислений играет большую роль в вопросе повышения производительности программных продуктов в современном мире. Многие широко используемые языки программирования, такие как Python добавляют в свои библиотеки материалы, позволяющие обрабатывать данные параллельно.

Проанализировав данные полученные в результате моделирования системы фазовых осцилляторов Курамото можно сказать, что система хорошо поддаётся распараллеливанию. Время вычислений уменьшается на $T/n + T_c$, где T – время выполнения распараллеленного участка на одном юните, T_c – постоянная поправка, учитывающая присутствие нераспараллеленных операций в программе, n – количество используемых вычислительных юнитов.

При использовании стандарта MPI для вычислений на CPU время выполнения растёт незначительно, до того пока количество осцилляторов не начинает отличатся от количества рабочих юнитов CPU больше чем на порядок. Стандарт OpenCL показывает лучшие результаты при увеличении количества осцилляторов т.к., несмотря на такое же количество виртуальных вычислительных юнитов (в настоящем эксперименте их было 8 и на CPU, и на

GPU), видеокарта (GPU) содержит большее количество физических процессоров, а значит может поддерживать большее количество параллельных потоков одновременно.