Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

**Институт информационных технологий**

Специальность «Инженерно-психологическое обеспечение информационных технологий»

**КОНТРОЛЬНАЯ РАБОТА**

По курсу «Интерфейс в системах информационных технологий»

Вариант 2

«Параллельные вычисления, как средство оптимизации выполнения программ»

Студентки 3 курса 680971 группы

заочного отделения

Барковской Ольги Вячеславовны

Зачетная книжка № 68097002

тел. +375(29) 141-14-74

:

Минск 2019

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc535160071)

[1. Параллельные вычисления, как средство оптимизации выполнения программ 4](#_Toc535160072)

[Вывод 9](#_Toc535160073)

[СПИСОК ЛИТЕРАТУРЫ 10](#_Toc535160074)

# Введение

Многие задачи требуют вычислений с большим количеством операций, которые занимают значительные ресурсы даже современной техники, более того, можно с уверенностью считать, что каких бы скоростей ни достигла вычислительная техника, всегда найдутся задачи, на решение которых потребовалось значительное время. Многие из таких сложных задач требуют, чтобы результат был получен за как можно меньшее время или даже строго ограниченное. К таким задачам, например, относится прогнозирование погоды, обработка изображений и распознание образов при управлении техникой. С другой стороны, представляет большую техническую проблему уменьшение времени исполнения каждой операции в микропроцессоре.

Очевидным способом увеличить скорость вычислений было бы применение не одного вычислительного устройства, а нескольких, работающих совместно над решением одной задачи. Такой подход носит название параллельных вычислений. Несмотря на кажущуюся простоту решения оно является подчас весьма нетривиальной задачей по проектированию вычислительной техники и разработки алгоритмов. Первая проблема кроется в том, что для того, чтобы задачу можно было решить с помощью параллельных вычислений алгоритм её решения должен допускать распараллеливание, мало того, далеко не каждая задача может быть решена параллельным алгоритмом. Другой же, не менее важной проблемой является построение системы, на которой бы возможна была реализация параллельных вычислений.

# 1. Параллельные вычисления, как средство оптимизации выполнения программ

Параллельные вычисления (параллельная обработка) – это использование нескольких или многих вычислительных устройств для одновременного выполнения разных частей одной программы (одного проекта).

Параллельные вычисления – такой способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно.

Параллельные вычисления – вычисления, которые можно реализовать на многопроцессорных системах с использованием возможности одновременного выполнения многих действий, порождаемых процессом решения одной или многих задач [одного проекта].

Существуют различные способы реализации параллельных вычислений. Например, каждый вычислительный процесс может быть реализован в виде процесса операционной системы, либо же вычислительные процессы могут представлять собой набор потоков выполнения внутри одного процесса ОС. Параллельные программы могут физически исполняться либо последовательно на единственном процессоре — перемежая по очереди шаги выполнения каждого вычислительного процесса, либо параллельно — выделяя каждому вычислительному процессу один или несколько процессоров (находящихся рядом или распределённых в компьютерную сеть).

Основная сложность при проектировании параллельных программ — обеспечить правильную последовательность взаимодействий между различными вычислительными процессами, а также координацию ресурсов, разделяемых между процессами.

Основная цель параллельных вычислений – уменьшение времени решения задачи. Многие необходимые для нужд практики задачи требуется решать в реальном времени или для их решения требуется очень большой объем вычислений[1].

Отметим, что увеличение числа процессоров не обязательно приводит к уменьшению времени решения задачи. (Если небольшую яму попытаются рыть одновременно 10 человек, то они будут только мешать друг другу.)

Использование параллельной обработки данных – не единственный путь увеличить скорость вычислений. Другой подход – увеличивать мощность процессорных устройств. Ограничениями такого подхода являются:

1. Ограниченность скорости переключения. Даже при самых быстрых коммуникациях – оптических – скорость переключения не может превышать скорость света.

2. Ограниченность размеров переключателей. Чем меньше размер компонентов устройства, тем быстрее устройство может работать. Однако существует физический предел на размер компонентов, что связано с их молекулярным и атомным строением.

3. Экономические ограничения. Для увеличения скорости процессора, плотности упаковки, числа слоев в кристалле приходится решать все усложняющиеся научные, инженерные, производственные проблемы. Вот почему каждое новое поколение процессоров дорого стоит.

Всегда найдутся большие задачи, для решения которых потребуются мощности параллельного компьютера. Задача параллельных вычислений – создание ресурса параллелизма (получение параллельного алгоритма) в процессах решения задач и управление реализацией этого параллелизма с целью достижения наибольшей эффективности использования многопроцессорной вычислительной техники.

Получить параллельный алгоритм решения задачи можно путем распараллеливания имеющегося последовательного алгоритма или путем 2 разработки нового параллельного алгоритма. Возможно, для осуществления распараллеливания алгоритм решения задачи придется заменить или модифицировать (например, устранить некоторые зависимости между операциями).

Существует два основных подхода к распараллеливанию вычислений в микропроцессорных системах, называемые однопоточным и многопоточным параллелизмом. Различие заключается в использовании одного или нескольких потоков исполнения для параллельных вычислений.

Однопоточный параллелизм заключается в параллельном выполнении операций внутри одного потока исполнения. Возможность однопоточного параллелизма определяется архитектурой микропроцессора, а конкретно его способностью считывать из памяти и исполнять одновременно несколько операций.

Однопоточный параллелизм обладает своими достоинствами и недостатками.

Достоинства:

* Отсутствие необходимости синхронизации — все операции выполняются внутри одного потока, и, следовательно, в строго определённой последовательности.
* Отсутствие необходимости поддержки паралеллизма на уровне операционной системы.
* Отсутствие необходимости в средствах управления разделяемыми ресурсами (арбитража).

Недостатки:

* Затруднённость использования в алгоритмах с условными переходами.
* Необходимость адаптации программы для эффективного использования ресурсов микропроцессора, например, при переходе с одной модели на другую.

Существуют следующие методы достижения параллельных вычислений:

* Упаковка данных для групповой обработки единичными инструкциями применением специальных методик. Например, возможно сложить две пары 8-битных данных 16-битной операцией исключив возможность переполнения. Метод имеет очень ограниченное применение.
* Суперскалярная архитектура. Устройство управления микропроцессора самостоятельно анализирует поток инструкций на предмет возможности параллельного исполнения и управляет несколькик функциональными устройствами.
* Векторная обработка. Микропроцессор имеет инструкции, производящие групповые однотипные операции. Однотипные операнды упаковываются в один векторный регистр. Этот метод аналогичен первому, но обеспечение параллельности лежит на микропроцессорной архитектуре. Векторные регистры как правило имеют большую разрядность. Требуется адаптировать программу для использования векторных инструкций или применять оптимизирующий компилятор.
* Микропроцессор с явным паралеллизмом. Метод аналогичен второму, но программа для такого процессора содержит явные указания на то, какие операции надо выполнять параллельно. Распараллеливание вычислений в данном случае полностью лежит на программисте или оптимизирующем компиляторе.

Многопоточный параллелизм — использование нескольких потоков для достижения параллельного исполнения операций. Для того, чтобы обеспечить многопоточный параллелизм необходимо создать систему с несколькими процессорами или процессорными ядрами.

Поэтому были разработаны специальные технологии для создания многопроцессорных систем. Которые позволяли обрабатывать данные параллельно, а, следовательно, быстрее. Соответственного создавались операционные системы поддерживающие многопроцессорные технологии. Такие как: Solaris (Sun Microsystems), Unix-подобные OS: Irix (SGI), AIX (IBM); Linux RedHat; Windows XP. Рассмотрим операционную системы Solaris версии 2.4. Solaris 2.4 - это Unix-подобная система, разработанная Sun Microsystems [2].

В операционной системе Solaris 2.4 существует такое понятие как поток. Поток (thread) — это последовательность инструкций выполняемых в пределах контекста процесса. Эта операционная система поддерживает многопоточные процессы. Слово «многопоточные» подразумевает содержание множества управляемых потоков. Традиционный UNIX процесс содержит один управляемый поток. Многопоточный в свою очередь содержит множество потоков, которые выполняются независимо. Так как каждый поток выполняется независимо, распараллеливание кода программы приводит к:

* Улучшению чувствительности приложения,
* Использование многопроцессорности более эффективно,
* Улучшает структуру программы,
* Использование меньше ресурсов системы,
* Улучшение представления.

Одновременность имеет место, когда не меньше двух потоков в процессе в одно время. Параллельность возникает, когда не меньше двух потоков выполняются одновременно. В многопоточном процессе на одном процессоре, процессор может распределять ресурсы между потоками, в результате получаем одновременное выполнение. В похожем многопоточном процессе на общей памяти в многопроцессорной системе, каждый поток в процессе может выполняться на отдельном процессоре в одно и то же время, в результате получаем параллельное выполнение. Когда процесс имеет столько же потоков, или меньше, сколько и процессоров, то система поддержки потоков и операционная система «уверены», что каждый поток исполняется на своём процессоре [3].

Потоки видны только внутри процесса, когда они разделяют ресурсы процесса такие, как адресное пространство, открытые файлы, и т. д. Каждая нить имеет уникальные ID, регистр, стек, маску, приоритет. Т.к. потоки делят инструкции процесса и большинство данных, изменения данных одним потоком видно другими потоками в процессе. Когда поток должен взаимодействовать с другими потоками процесса, он может сделать это без привлечения операционной системы. Потоки это главный программный интерфейс в многопоточном программировании. Потоки пользовательского уровня управляются в пользовательском пространстве и поэтому могут запрещать контекстному ядру переключение ресурсов. Приложение может иметь тысячи потоков, и не потреблять много ресурсов ядра. Количество ресурсов ядра, потребляемых приложением, во многом определяется самим приложением. По умолчанию потоки «легковесны». Для получения большего контроля за потоками, приложение может ограничивать потоки. Когда приложение ограничивает потоки в доступе к ресурсам, поток становится ресурсами ядра. Функции для работы с потоками, такие как thr\_create(….), thr\_self(), thr\_join(….) и т. д., описаны в библиотеке libthread. Функция thr\_create создает поток в зависимости от заданных параметров. Функция thr\_join объединяет потоки в параллельный процесс. Функция thr\_self возвращает номер потока в процессе [4].

# Вывод

Использование большего числа процессоров ускоряет работу программы и не сильно усложняет работу программистов. Следовательно, при работе с большим количеством данных рациональнее использовать многопроцессорные системы.

Принципы организации однопоточного и многопоточного параллелизма, их особенности, достоинства и недостатки очень различны, и имеют мало общего как в реализации вычислительной системы, так и в построении программного обеспечения.

# 

# СПИСОК ЛИТЕРАТУРЫ

1. Словарь по кибернетике / Под редакцией академика В. С. Михалевича. — 2-е. — Киев: Главная редакция Украинской Советской Энциклопедии имени М. П. Бажана, 1989. — 751 с. — (С48). — 50 000 экз. — ISBN 5-88500-008-5.
2. RS/6000 SP: Practical MPI Programming. — IBM RedBook, 1999. — 238 с. Архивировано 19 января 2008 года. (англ.)
3. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб: БХВ-Петербург, 2002. — 608 с. — ISBN 5-94157-160-7.
4. Оленев Н. Н. Основы параллельного программирования в системе MPI. — М.: ВЦ РАН, 2005. — 80 с. — ISBN 5201098320