30. ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНОГО ВЫПОЛНЕНИЯ КОДА

Панизник А.С, студент гр. 272303, Липницкая Н.И., ассистент кафедры ЭИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Ефремов А.А. – канд. эк. наук, доцент кафедры ЭИ

Аннотация. Данная работа представляет собой анализ технологии параллельного выполнения кода, в процессе анализа будут рассмотрены все особенности, преимущества и примеры использования параллельного выполнения кода. Описание методов параллельной обработки позволит оценить уровень развития технологии и предугадать дальнейшие перспективы её усовершенствования.

Ключевые слова. Параллельное программирование, узлы, ядра, интерфейс передачи сообщений, мелкозернистые и крупнозернистые типы параллельных процессов, многоядерные процессоры, открытая многопроцессорная обработка, операционная система, центральные процессоры, графические процессоры, параллельные алгоритмы, код.

Сейчас практически невозможно найти современную компьютерную систему без многоядерного процессора. Даже недорогие мобильные телефоны предлагают наличие пары ядер. Идея многоядерных систем проста: это относительно эффективная технология для масштабирования потенциальной производительности процессора. Эта технология начала использоваться около 20 лет назад, и теперь каждый современный разработчик способен создать приложение с параллельным выполнением для использования такой системы.

Параллельное программирование используется, когда пользователь хочет быстро обработать большие объемы данных. Данный способ написания кода помогает завершить проекты быстро и эффективно. Параллельное программирование имеет схожее значение и иногда заменяется терминами параллельная обработка, параллельные вычисления и решения для параллельных вычислений.

Параллельное программирование работает путем назначения задач разным узлам или ядрам. В системах высокопроизводительных вычислений (HPC) узел представляет собой автономную единицу компьютерной системы, содержащую память и процессоры, работающие под управлением операционной системы. Процессоры, такие как центральные процессоры (ЦП) и графические процессоры (ГП), представляют собой микросхемы, содержащие набор ядер. Ядра — это модули, выполняющие команды; может быть несколько ядер в процессоре и несколько процессоров в узле [1].

Многоядерные процессоры, часто встречающиеся в современных компьютерах, и любая система с более чем одним процессором способны выполнять параллельную обработку.

Методы параллельной обработки можно использовать на встроенных, мобильных устройствах, ноутбуках и рабочих станциях, и крупнейших в мире суперкомпьютерах. Разные языки программирования используют разные технологии для обеспечения принципа параллельности. Открытая многопроцессорная обработка предоставляет межплатформенный API для разработки параллельных приложений с использованием языков программирования C, C++ и Fortran для ядер одного процессора.

Интерфейс передачи сообщений (MPI) позволяет выполнять параллельные процессы между разными компьютерами или узлами.

Существует несколько типов параллельной обработки, такие как MMP, SIMD, MISD, SISD и MIMD, из которых SIMD, вероятно, является наиболее популярным. Наиболее часто используются такие типы как SIMD и MIMD [2].

Когда обработка выполняется параллельно, большая работа разбивается на несколько более мелких задач, которые лучше подходят для количества, размера и типа доступных процессорных единиц. После разделения задачи каждый процессор начинает работать над своей частью. Чтобы оставаться на связи друг с другом и узнавать, как продвигаются их задачи, разработчики используют программное обеспечение. После обработки всех частей программы результатом является полностью обработанный сегмент программы. Он не зависит от того, было ли число процессоров и задач и процессоров равными и все они завершились одновременно или один за другим. Различают два типа параллельных процессов: мелкозернистые и крупнозернистые. Зернистость – это мера отношения количества вычислений, сделанных параллельной задаче, к количеству пересылок данных. Мелкозернистый параллелизм – очень мало вычислений на каждую пересылку данных. Крупнозернистый параллелизм – интенсивные вычисления на каждую пересылку данных (данные пересылаются большими порциями) [3].

В настоящее время параллельная обработка или параллельные вычисления имеют много важных применений. Поскольку параллельное программирование отлично подходит для декомпозиции сложных задач, оно обычно лучше всего проявляется при использовании сложных вычислений, больших наборов данных или больших симуляций. Примерами являются суперкомпьютеры для использования в астрономии, прогнозирование в сельском хозяйстве, расчёты рисков и криптовалюты в банковском деле, точная медицинская визуализация, ноутбуки и компьютеры, продвинутая графика в индустрии развлечений, прикладная физика, климатические исследования, электротехника, финансово-экономическое моделирование, молекулярное моделирование, национальная оборона и ядерное оружие, разведка нефти и газа, квантовая механика.

Операционные системы и язык C++ предоставляют интерфейсы для создания потоков, которые потенциально могут выполнять один и тот же или различные наборы инструкций одновременно. Именно поэтому в данный момент параллельное выполнение кода имеет широкое распространение, не смотря на некоторые трудности при его реализации. При параллельном

выполнении кода в языке C++ возникают некоторые проблемы. Их основными источниками являются data races (гонки данных) и race conditions (состояние гонки). Для решения возникающих проблем используются различные инструменты, такие как атомарные операции, мьютекс.

Многопоточное программирование — это подмножество параллельного программирования, в котором одновременно выполняется более одного набора последовательных инструкций («потоков»). Многопоточность — это концепция, которая может существовать либо на одном ядре, либо на нескольких процессах [4].

Существуют специализированные параллельные языки и расширения существующих языков. Наиболее распространёнными являются HOPMA, ABCL, Adl и Ada [5].

Наиболее значительным преимуществом параллельного программирования является более быстрое выполнение кода, экономя время выполнения и усилия. Вместо запуска последовательного кода запускается параллельный код. Эти преимущества особенно очевидны при крупномасштабном параллелизме данных, так как процессы там более сложные, и их разбиение на более мелкие подпроцессы является необходимостью. При параллелизме данных каждый поток работает над одним и тем же набором задач, над подмножеством значений. Это означает, что каждый поток выполняет одну и ту же задачу с разными наборами данных: распараллеливаются сами данные, а не сами задачи. Меньшее количество задач означает меньше времени и усилий, а это значит, что больше времени можно потратить на другие детали и проекты.

Однако параллельное программирование не ограничивается параллелизмом данных. Мы можем распределить выполнение кода между несколькими задачами для более быстрого выполнения, распределяя задачи по разным потокам и по разным процессорам. Поступая таким образом, мы также увеличиваем естественные ресурсы программы для работы и тем самым увеличиваем ее возможности, делаем многие вещи быстрее.

В чём заключается отличие между параллельным и последовательным выполнением кода? Параллельная обработка выполняет несколько задач одновременно разными процессорами. Последовательное программирование или последовательная обработка — это тип программирования, который выполняет одно задание за раз и выполняет все это на одном процессоре в последовательности. Этот тип программирования обычно требует больше времени для выполнения функций по сравнению с методами параллельной обработки. Параллельное программирование приводит к ряду проблем, которые невозможно наблюдать в последовательной программе. Более того, эти проблемы не всегда легко обнаружить, и они не всегда очевидны. Используя параллельное программирование в языке C++ существуют такие библиотеки, как oneTBB, которые упрощают параллельное программирование во многих аспектах [6].

В результате анализа технологии параллельного кода можно сделать вывод о том, что параллельное программирование – это эффективный и выгодный способ реализации крупных задач на производствах. Оно позволяет значительно уменьшить время, затраченное на выполнение проектов. Чтобы данный процесс был более выгодным, необходимо составить подробную инструкцию для команды разработчиков, чтобы в процессе реализации все детали были понятны, а последовательность действий была чёткой и лаконичной. Хотя параллельное программирование может создавать технический долг и требовать больших затрат времени на настройку процесса — программистам необходимо разрабатывать эффективные параллельные алгоритмы и код — этот процесс в целом экономит время. Используя мощность параллельной обработки, параллельное программирование запускает определенную программу на нескольких вычислительных узлах и ядрах ЦП одновременно, это помогает значительно уменьшить количество времени, отведённого на этап разработки. Обработка данных не должна быть сложной, а с помощью параллельного программирования разработчик может вывести свой список дел на новый уровень.

Список использованных источников:

1. Гафаров Ф.М. Г12 Параллельные вычисления: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 149 с.

2. Лупин С.А., Посыпкин М.А. Технологии параллельного программирования. Серия: Высшее образование. - М.: Форум, Инфра-М, 2008. - 208 с.

3. Миллер Р., Боксер Л. Последовательные и параллельные алгоритмы: Общий подход. - М.: БИНОМ. Лаборатория знаний, 2006. - 406 с.

4. .NET | Параллельное программирование - Professor Web [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/csharp/thread_and_files/level2/2_1.php – Дата доступа: 01.04.2023.

5. Параллельные языки и расширения существующих языков [Электронный ресурс]. – Режим доступа: https://parallel.ru/tech/tech_dev/par_lang.html – Дата доступа: 01.04.2023.

6. Разбираемся с параллельными и конкурентными вычислениями в Python [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/wunderfund/articles/581994/ – Дата доступа: 01.04.2023.