ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА ПРИ ВЫПОЛНЕНИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В СИСТЕМЕ FLC2

Логинова И. П.
Объединённый институт проблем информатики Национальной академии наук Беларуси
Минск, Республика Беларусь
E-mail: irilog@mail.ru

Ввод-вывод – важный фактор при проведении высокопроизводительных вычислений в современных многопроцессорных системах. Организация ввода-вывода при параллельных вычислениях является серьезной задачей, как с точки зрения эффективности, так и с точки зрения мобильности и масштабируемости параллельной программы. Рассмотрены некоторые аспекты реализации операций ввода-вывода в системе FLC2 в процессе выполнения процедур логической оптимизации при проведении параллельных вычислений на многоядерной вычислительной системе.

Введение

Функционально операции ввод-вывода в программах можно разделить на такие группы:

1. Явные операторы ввода-вывода, которые необходимы для ввода начальных данных и вывода окончательных результатов.
2. Для задач с большим временем выполнения необходимо периодически запоминать состояние задачи, так называемые контрольные точки.
3. В задачах требуется запоминать текущее состояние вычислений через определенные промежутки времени, например, для использования в подсистемах визуализации.
4. Вычисления с большими массивами данных, расположенными на внешней памяти.

Разные функциональные назначения операций ввода-вывода требуют использования разных подходов к их реализации. Неэффективное управление вводом-выводом при проведении параллельных вычислений может свести на нет любые оптимизации параллельного выполнения задач, если в них имеется интенсивный ввод-вывод.

I. Параллельные вычисления в FLC2

Функционал системы логической оптимизации функционально-структурных описаний дискретных устройств FLC2 [1] позволяет реализовать достаточно простой подход к организации параллельных вычислений на многоядерной вычислительной системе. В основе лежит работа параллельных процессов, каждый из которых выполняется на ядре процессора, представляя работу одного потока в одном процессе [2]. В отдельном процессе производится старт команды, аргументы которой определяют имя внешней программы и исходный объект оптимизации. Исходные объекты исполняемой программы являются независимыми друг от друга. Организовать совокупности независимых объектов для проведения параллельных вычислений с использованием программ технологически независимой оптимизации позволяет среда системы FLC2. Средствами FLC2 можно проводить проектные операции для преобразования исходного объекта в одноранговую сеть независимых объектов и осуществлять параллельные вычисления посредством запуска процессов, в которых выполняется программа для одного элемента сети. Для такой организации параллельных вычислений был использован стандарт OpenMP (версия 3.5 и выше), который специфицирует набор директив компилятора (для языков C, C++ и др.), функции библиотек и переменных окружения [3].

II. Обзор существующих решений

В настоящее время нет общего широко используемого интерфейса параллельного ввода–вывода (далее в/в). Для различных технологий параллельного программирования, существуют такие решения [3], как, например, MPI-IO или OpenMP-IO. Как правило, при реализации различных параллельных программ используются специализированные библиотеки в/в, предназначенные для работы с научными данными различных форматов. Отсутствие общего подхода и стремление к достижению максимальной производительности в/в привело к тому, что разработчики новых языков программирования, вынуждены разрабатывать собственные средства для параллельного в/в. Для приложений, использующих OpenMP, возможности в/в существенно ограничены, следующими схемами организации в/в, когда N процессов пишут в один файл или N процессов пишут в N файлов. Для OpenMP можно использовать следующие подходы. Например, при использовании совместно используемого файла для поддержки согласованности файла с результатом, операции чтения и записи должны выполняться вне параллельных областей [4]. В случае, если несколько процессов обращаются к одному файлу, доступ к дескриптору файла (общему для всех потоков) должен быть защищен, например, с использованием механизма критических секций. Другой подход – позволить каждому процессу использовать отдельный файл, чтобы избежать ошибок типа гонок (race condition) или ошибок синхронизации при доступе к общему дескрипто-

ру файла. Хотя такой подход часто приводит к более высокой производительности, он имеет следующий недостаток: он требует дополнительных (достаточно дорогостоящих) шагов пред- и постобработки, для того чтобы создать необходимое количество входных файлов и объединить выходные файлы, записанные разными процессами. Для повышения производительности операций в/в в OpenMP приложениях разрабатываются интерфейсы параллельного ввода-вывода, который обеспечивает доступ нескольких потоков к одному файлу без необходимости явного использования операций синхронизации [3,4]. Эти спецификации очень близки к спецификациям MPI-IO. Основное отличие от MPI-IO – отсутствие неблокирующих операций ввода-вывода.

III. Схема обработки вызовов ввода-вывода в параллельных процессах

Общепринято, что процессы рассматриваются операционной системой (ОС) как заявки или контейнеры для всех видов ресурсов, кроме процессорного времени. Этот ресурс распределяется ОС между другими единицами работы – потоками, которые представляют собой последовательности команд. В простейшем случае процесс состоит из одного потока. Новый процесс может быть создан по запросу текущего процесса. Ресурсы могут быть выделены процессу на все время его жизни или только на определенный период. За обеспечение процессов необходимыми ресурсами отвечает подсистема управления процессами и потоками. Создание процесса включает загрузку кодов и данных исполняемой программы данного процесса в оперативную память. Когда процесс завершается благодаря одному из событий, например, обычного завершения выполняемой в процессе программы подсистема управления процессами закрывает все файлы, с которыми работал процесс, освобождает области оперативной памяти, отведенные под коды, данные и системные информационные структуры процесса.

Стандартные средства современных ОС не позволяют создать для одного приложения несколько процессов для параллельной работы, но многоядерность и использование технологии OpenMP [4] позволили реализовать подход [2]. При этом подходе процесс, выполняемый в каждом ядре, сам решает, когда ему отдать управление кооперативному (non-preemptive) планировщику задач, чтобы последний выбрал из пула задач готовый к выполнению процесс, используя механизм work stealing. Процесс работает в пользовательском режиме, но когда он обращается к системному вызову в/в, то ввод-вывод осуществляется одновременно с вычислениями в ядрах процессора, которые «отвлекаются» только для выдачи команд контроллерам о начале и завершении ввода-вывода. ОС может выполнять вызовы в/в в синхронном и асинхронном режимах.

В первом случае, как представлено на схеме на рис. 1, процесс, сделавший такой вызов, приостанавливается до тех пор, пока контроллер в/в не выполнит свою работу. Затем планировщик переводит процесс в состояние продолжения вычислений, процесс может воспользоваться результатами вызова в/в. Синхронный в/в выполняется следующим образом. В каждым процессе, выполняемом на ядре многопроцессорной системы, для ввода исходных данных и вывода результата открываются отдельные файлы, что является достаточно удобным средством организации параллельной работы контроллеров ввода-вывода.

Асинхронный вызов в/в не приводит к переводу процесса в режим ожидания и после выполнения некоторых начальных системных действий, например запуска операции ввода-вывода, управление возвращается вызывающему процессу. Асинхронный системный вызов в подходе [2] используется для получения трассы выполнения параллельных процессов и визуализации.

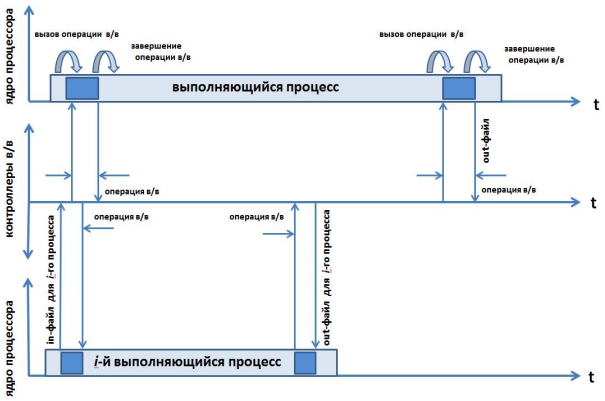

Рис. 1 – Схема синхронного выполнения операций ввода вывода в системе FLC2 при работе параллельных процессов

IV. Список литературы

1. Бибило, П. Н. Система логической оптимизации функционально-структурных описаний цифровых устройств на основе продукционно-фреймовой модели представления знаний /П. Н. Бибило, В. И. Романов // Проблемы разработки перспективных микро- и наноэлектронных систем – (МЭС-2020) : IX Юбилейная Всероссийская научно-техническая конференция : Сборник трудов ; под общей ред. А. Л. Стемпковского. – Москва : ИППМ РАН, 2021 Москва : ИППМ РАН, 2020. – С. 9-16.
2. Бибило, П. Н. Экспериментальное сравнение эффективности программ минимизации систем булевых функций в классе дизъюнктивных нормальных форм / П. Н. Бибило, И. П. Логинова // Информатика. – 2022. – Т. 19, № 2. – С. 26-55.
3. Бахтин, В. А. Отладка параллельных программ в DVM-системе / В. А. Бахтин, Д. А. Захаров, А. А. Ермичев, В. А. Крюков // Электронные библиотеки // – 2020. – Vol. 23, № 4. – С. 866–886.
4. Kshitij Mehta, Edgar Gabriel, Barbara Chapman. Specification and Performance Evaluation of Parallel I/O Interfaces for OpenMP. OpenMP in a Heterogeneous World. Lecture Notes in Computer Science Volume 7312, 2012, pp. 1-14.