ДИАГРАММА ГАНТА КАК СРЕДСТВО ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

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

В работе представлен подход, предназначенный для применения таких инструментов визуализации как диаграммы Ганта при анализе производительности параллельных решений различных задач логической оптимизации. Подход может применяться для поиска характеристик исходных объектов, которые приводят к получению эффективного решения логико-комбинаторных задач оптимизации.

Введение

Решение некоторых задач проектирования СБИС сверхбольших размерностей невозможно без использования масштабируемых параллельных алгоритмов, ориентированных на многопроцессорные/многоядерные вычислительные системы. Большая часть алгоритмов из области логического проектирования имеют свои программные реализации, которые используются в составе различных САПР. Как правило, эти программы, представлены последовательно исполняемым кодом без привязки к архитектуре используемых вычислительных средств. В последние годы были приложены определенные усилия по модернизации существующих алгоритмов задач логического проектирования и разработке новых, параллельных реализаций этих алгоритмов. Результатом проведенных исследований стало признание того, что большинство логико-комбинаторных алгоритмов не допускают эффективного распараллеливания.

I. параллельный запуск программ

Один из вариантов организации распараллеливания для программ проектирования (включенных в состав некоторых САПР) для многоядерной системы с общей памятью на основе стандарта OpenMP предложен в [1]. OpenMP является открытым стандартом для реализации многопоточных приложений, содержит набор различных директив компилятора, функций соответствующей библиотеки и переменных окружения. Стандарт OpenMP можно рассматривать как высокоуровневую надстройку над POSIX или Windows Threads (библиотеками потоков). При проектировании параллельных программ требуется выполнение следующих действий: разделение (декомпозиция) объекта проектирования на части, выявление информационных зависимостей между частями и распределение работы с этими частями по вычислительным элементам (в системах с общей памятью - по ядрам процессора). В основу подхода [1] положен запуск исполняемых модулей программ и организация их параллельной работы. Поскольку любая программа, загруженная в память компьютера для выполнения, образует процесс, то программа, реализующая запуск нескольких процессов, будет работать быстрее, если организовать параллельное выполнение запущенных процессов. Это может быть осуществлено путем вставки директив OpenMP в код программы, запускающей в цикле выполнение нескольких процессов. Такую параллельную работу с программами можно реализовать, если: 1) выполняется множество независимых процессов; 2) процессам не надо общаться; 3) результаты, полученные процессами, сводятся в один когда завершат работу все процессы.

II. архитектура программы для организации параллельной работы

Схема реализации параллельных вычислений такова: в головной программе запускается множество параллельных задач (процессов); эти задачи динамическим образом распределяются по всем ядрам (включая логические процессоры, если они есть); каждая задача определяет работу одного процесса. У всех процессов свои адресные пространства, процессы взаимодействует с памятью независимо, каждый процесс работает со своими входными данными. Когда головная программа производит запуск другой программы, создается новый (дочерний) процесс. Работа дочернего процесса может осуществляться в синхроном и асинхронном режиме. Размещение директивы pragma omp parallel for перед циклом, который запускает группу дочерних процессов, распределяет эти процессы по параллельным регионам -–каждый процесс в отдельный параллельный регион. Все потоки в параллельных регионах выполняют, как правило, один код. Этот код представлен командой start, которая производит запуск консольной реализации какой-либо внешней программы [1]. Аргументами команды start являются параметры командной строки запускаемой программы. В конце параллельной области происходит неявная барьерная синхронизация, т.е. все потоки останавливаются до тех пор, пока последний поток не выполнит свою программу целиком. Но при вставке в цикл, директивы pragma omp task синхронизация потоков отменяется, создается набор задач (task), число задач определяется числом процессов, все задачи помещаются в оче-

редь. Из очереди задачу (процесс) может взять один из рабочих потоков для выполнения. Задач может быть намного больше числа рабочих потоков. Распределение задач по потокам осуществляется планировщиком задач, принцип работы которого представлен на рисунке 1. Планировщик реализует формирование очереди, проводит распределение задач по рабочим потокам, применяя стратегию work-stealing. Суть этой стратегии заключается в том, что освободившийся рабочий поток при отсутствии ожидающих задач в своей очереди заимствует задачу у одного из рабочих потоков.

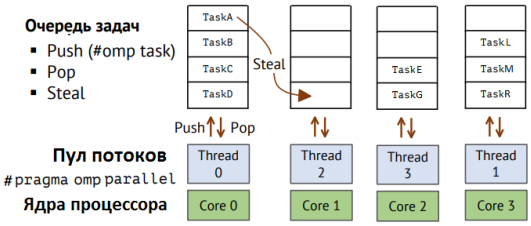

Рис. 1 – Распределение очереди в пуле потоков

III. Организация визуализации

Для оценки производительности параллельных вычислений следует провести оценку времени выполнения каждого процесса - получить времена начала и завершения дочерних процессов. Поэтому в тело цикла запуска процессов вводятся так называемые критические секции (pragma omp critical), которые содержат код для вычисления и сохранения в трассировочном файле характеристик каждого процесса в порядке наступлений событий в процессе и рабочем потоке. Псевдокод организации работы параллельных процессов приведен на рисунке 2.

```
#pragma omp parallel for
// цикл по числу NX запусков внешней программы
for(i=0; i<NX; i++) {
# pragma omp task .....
 {
  // вычисление времени начала задачи
  # pragma omp critical
   {  ...  }
   start(process); // запуск внешней программы
  // вычисление времени завершения задачи
  # pragma omp critical
   {  ...  }
} }
```

Рис. 2 – Псевдокод параллельного пуска процессов

После завершения вычислений проводится упорядочение (с использованием алгоритмов рекурсивной сортировки) данных трассировочного файла. В целом, даже в упорядоченном виде данные из трассировочного файла - время, идентификаторы задачи, потока, имя программы, достаточно тяжелы для усваивания этой информации, поэтому удобным инструментом при анализе результатов работы параллельной программы является визуализация упорядоченных данных в виде диаграмм Ганта. Обычно, как инструмент визуализации, диаграмма Ганта используется при планировании проектов и представляет из себя столбчатую диаграмму с накоплением, которая показывает даты начала, окончания, сроки проекта. Использование этого инструмента в контексте организации параллельных вычислений дает визуальное представление о времени выполнении задач. На рисунке 3 приведен вид диаграммы Ганта, полученной в результате следующего эксперимента. Было проведено параллельное выполнение трех разных программ логической оптимизации, каждая программа работала со своим объектом проектирования, каждый объект, был разбит на группы независимых частей с использованием методов декомпозии - в общей сложности, параллельно выполнялись 84 задачи оптимизации.

Рис. 3 – Визуализация результатов программы

Данная диаграмма отображает распределение по 16-ти процессорам этих задач. По диаграмме видно, что время выполнения отдельных задач существенно различается, что обусловлено разными алгоритмами оптимизации и разными размерами исходного объекта для каждой задачи. В итоге, анализ упорядоченных данных трассировочного файла совместно с визуальным представлением этих данных в виде диаграмм позволяет оценить влияние различных методов декомпозиции объектов проектирования, а также сравнить быстродействие программ оптимизации. Таким образом, диаграмма Ганта представляется удобным средством визуально-структурированного представления данных при анализе результатов параллельной работы программ проектирования. Программа визуализаци диаграмм Ганта реализована при использовании кроссплатформенной библиотеки QtCharts (включена в Qt5.7).

1. 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.