НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Архитектура вычислительных систем.
Пояснительная записка.
Практическое задание 5
Вариант 22.

Исполнитель
Студент БПИ205
Никитин Никита Евгеньевич

Описание полученного задания

Задача об инвентаризации по книгам. После нового года в библиотеке университета обнаружилась пропажа каталога. После поиска и наказания, виноватых ректор дал указание восстановить каталог силами студентов. Фонд библиотека представляет собой прямоугольное помещение, в котором находится M рядов по N шкафов по K книг в каждом шкафу. Требуется создать многопоточное приложение, составляющее каталог. При решении задачи использовать метод «портфель задач», причем в качестве отдельной задачи задается внесение в каталог записи об отдельной книге.

Описание подхода к решению задачи

Создан класс книги, содержащий основную информацию об экземпляре.
Также создан класс библиотеки, хранящий случайно-сгенерированные книги в трехмерном массиве.
В функции main() задается число студентов (потоков) и размерность полок (ряды/шкафы/кол-во в шкафу).
Функция work() потока работает, пока активен флаг о незавершенности работы. В каждой итерации ее цикла она запрашивает новую координату книги, которую следует обработать.
За выдачу координаты отвечает функция giveTask(). Выдача координаты в ней происходит потокобезопасно, при помощи mutex.
После получения новой задачи, поток считывает данные о книге, вносит ее в раздел каталога по первой букве автора, вносит ее в соответствующий раздел (один из двадцати шести) и сортирует этот раздел. Каталог разделен по первым буквам авторов, книги отсортированы по возрастанию имен авторов, далее по номеру ISBN и по названию.
Результат выводится в текстовый документ out.txt. В консоль выводится статистическая информация о работе потоков: номер потока, кол-во записанных книг, время простоя в ожидании разблокировки соответствующего раздела каталога.

Общие характеристики программы

Число заголовочных модулей — 2
Число модулей реализации — 3
Общий размер исходных кодов — 264

Время работы программы на тестовых данных

Каждый тестовый прогон выполнялся на библиотеке размерности 10*10*100.
Тестирование провожу на 3 конфигурациях:

1. MacOS Monterey, Apple M1(ARM): 8 ядер (4 производительных и 4 энергоэффективных).
2. Linux Mint 19, AMD A9-9410: 2 ядра.

В тесте Apple M1 изменяю число потоков с 1 до 8.
В тесте 2-ядерного маломощного AMD тестирую 1-4 потока.
Этого достаточно, чтобы сделать дельнейшие выводы. Привожу время работы и время простоя потока в ожидании разблокировки нужного ресурса.

Кол-во потоков	Время, сек/средний простой потока, сек	
	Apple M1	AMD A9-9410
1	23,7/0	45/0
2	14,2/2	27/3
3	11,8/5	35/8
4	10,3/8	48/13
5	9,86/11	
6	9,86/13	
7	9,6/14	
8	9,6/16	

Отмечу, что при прогонах на Apple M1 были загружены только производительные ядра. Энергоэффективные простаивали.

Первый вывод заключается в том, что при работе с потоками многое зависит от операционной системы и ее планировщика задач. Несмотря на оснащенность 8 ядрами, операционная система может ограничивать работу приложения исходя из собственных инструкций.

Второй вывод заключается в том, что эффективность программы растет до момента сравнивания числа потоков программы с числом ядер/потоков компьютера. Время простоя потоков в ожидании освобождения критических участков кода также увеличивается, однако окупается общим ускорением работы программы.

Третье: создание числа потоков, превышающего число вычислительных ядер, не дает ускорения работы программы, так как приводит к псевдопараллельному выполнению задач. Фактически одно и

то же ядро чередует исполнение кода разных потоков. В этом случае ответственность за эффективность лежит на системном планировщике, однако реального прироста достичь не удастся. Более того, в случае с маломощным процессором AMD, генерация и обслуживание лишних потоков обходится дорого и может привести к ухудшению производительности программы.

Многопоточность является важным аспектом эффективности работы программы. И если еще 9 лет назад процессор FX-8300 со своими 8 ядрами казался избыточным для домашнего пк, то сегодня большое число ядер становится нормой. А серверные процессоры давно обладают невообразимо большим числом процессоров. Поэтому при разработке ПО крайне важно обращать внимание и на этот аспект.