Практическое занятие №2

Разработка базы данных тестовых примеров задач параллельного программирования

(параллельная обработка экспериментальных данных)

Содержание

Введение. Парадигмы параллельного программирования

Парадигма программирования – способ концептуализации, который определяет, как следует проводить вычисления, и как работа, выполняемая компьютером, должна быть структурирована и организована.

Парадигмы параллельного программирования , подразделяются на следующие основные виды:

итеративный параллелизм;

рекурсивный параллелизм;

«клиенты и серверы»;

«производители и потребители»;

взаимодействующие равные.

Итеративный параллелизм – процессы выполняют циклические вычисления, решая одну задачу (итерации одного цикла). Чаще всего встречается в вычислениях, выполняемых на нескольких процессорах.

Рекурсивный параллелизм может использоваться, когда в программе есть одна или несколько рекурсивных процедур (функций), и их вызовы независимы, т.е. каждый из них работает над своей частью общих данных. Рекурсивный параллелизм используется для решения таких комбинаторных проблем, как сортировка, планирование (задача коммивояжера) и игры (шахматы и другие).

1. Парадигма параллельного программирования «клиенты и серверы»

Клиенты и серверы – наиболее распространенная модель взаимодействия в распределенных системах, от локальных сетей до всемирной сети Интернет. В этой модели реализуется две части: клиент и сервер.

Чаще всего они располагаются на разных машинах, и обычно один сервер обслуживает многих клиентов. Клиент отправляет запрос серверу и ждет ответа. Сервер ожидает запросов от клиентов, а затем действует в соответствии с этими запросами и возвращает ответ клиенту.

Иногда клиент может «исполнить» роль сервера, если сам будет получать запросы. Аналогично сервер будет выступать в роли клиента, если ему потребуется обращаться с запросами к другим программам.

Отношения между клиентом и сервером аналогичны отношениям в параллельном программирования между программой, вызывающей подпрограмму, и самой подпрограммой в последовательном программировании. Более того, как подпрограмма может быть вызвана из нескольких мест программы, так и у сервера обычно есть много клиентов.

Запросы каждого клиента должны обрабатываться независимо, однако параллельно может обрабатываться несколько запросов, подобно тому, как

одновременно могут быть активны несколько вызовов одной и той же процедуры.

Сервер может быть реализован как одиночный процесс, который не может обрабатывать одновременно несколько клиентских запросов, или (при необходимости параллельного обслуживания запросов) как многопоточная программа.

Клиенты обычно представляют собой множество потоков.

Одна из классических «задач о спящем парикмахере» описывает отношение в системах «клиент-сервер». У парикмахера есть одно рабочее место и приемная с несколькими стульями. Когда парикмахер заканчивает подстригать клиента, он отпускает клиента и затем идет в приёмную, чтобы посмотреть, есть ли там ожидающие клиенты. Если они есть, он приглашает одного из них и стрижет его. Если ждущих клиентов нет, он возвращается к своему креслу и спит в нем.

Каждый приходящий клиент смотрит на то, что делает парикмахер. Если парикмахер спит, то клиент будит его и садится в кресло. Если парикмахер работает, то клиент идет в приёмную. Если в приёмной есть свободный стул, клиент садится и ждёт своей очереди. Если свободного стула нет, то клиент уходит.

Клиентами являются, как уже можно догадаться, посетители салона, а парикмахера можно интерпретировать как сервер. Потоки-клиенты отдают запросы серверу, чтобы занять рабочее кресло, сервер получает и обрабатывает поступающие запросы. То есть он возвращает ответ, занят ли ресурс (кресло) или он готов обработать следующий запрос клиента.

Клиенты образуют очередь, которая является ограниченной.

Посетители уходят, если нет свободных кресел в зале ожидания. То есть придется создавать и отправлять на выполнение новые потоки (потоки-клиенты), что ограничивает применение описанной структуры.

2. Парадигма параллельного программирования «производители и потребители»

Производители и потребители — это взаимодействующие процессы. Они часто организуются в конвейер, через который проходит информация.

Прежде чем переходить к организации конвейеров, определим, что процесс-производитель генерирует в некотором буфере информацию, которая используется процессом-потребителем.

Здесь возникают проблемы переполнения и исчерпания буфера. Тогда при переполнении буфера производитель должен будет ждать, пока в буфере не освободится хотя бы один элемент, а при исчерпании буфера должен будет ждать потребитель, пока хотя бы один новый элемент не появится в буфере.

Как было сказано ранее, часто производители и потребители объединены в конвейер – последовательность процессов, в которой каждый потребляет данные предшественника и поставляет данные для последующего процесса.

Недостаток данной модели взаимодействия процессов: ограниченность буфера, необходимо всегда проверять размер буфера, иначе возникает опасность переполнения или истощения буфера.

3. Парадигма параллельного программирования “взаимодействующие равные”

Взаимодействующие равные – модель, в которой исключен не занимающийся непосредственными вычислениями управляющий поток.

Распределение работ в таком приложении либо фиксировано заранее, либо динамически определяется во время выполнения.

Одним из распространенных способов динамического распределения работ является «портфель задач». Портфель задач, как правило, реализуется с

помощью разделяемой переменной, доступ к которой в один момент времени имеет только один процесс.

Вычислительная задача делится на конечное число подзадач. Как правило, каждая подзадача должна выполнить однотипные действия над разными данными. Подзадачи нумеруются, и каждому номеру определяется функция, которая однозначно отражает номер задачи на соответствующий ему набор данных. Создается переменная, которую следует выполнять следующей. Каждый поток сначала обращается к портфелю задач для выяснения текущего номера задачи, после этого увеличивает его, потом берет соответствующие данные и выполняет задачу, затем обращается к портфелю задач для выяснения следующего номера задачи.

Естественно должен быть предусмотрен механизм остановки процессов при исчерпывании всего множества задач, как в «производителях и потребителях».

То есть поток получает задачу из портфеля и пока задача остается не выполненной, поток ее решает, а затем снова получает задачу из портфеля.

Рассмотрим задачу вычисления произведения матриц. Подзадачей будет вычисление строк результирующей матрицы, а портфель задач – переменная, в которую будем считать строки. При получении задачи, происходит считывание значения счетчика и его увеличение на 1. Когда счетчик будет равен размерности матрицы, вычисления заканчиваются. То есть поток без прерывания остальных возьмет задачу, при этом выполнив увеличение счетчика.

Таким образом, процессы работают независимо, каждый со своей скоростью, синхронизация происходит с помощью портфеля задач.

Проблема реализации этого алгоритма в том, что доступ к портфелю задач должен быть безопасным. Если между взятием новой задачи и увеличением счетчика работа выполняющего их потока прервется, то некоторую задачу, возможно, отработают несколько потоков.

4. Задание к практическому занятию

Исходные данные:

- массив результатов экспериментального исследования технического объекта (асинхронное скачкообразное изменение некоторого параметра)

- каждый эксперимент имеет графическое представление (*.jpg) и файл данных в формате *.csv

Требуется

1. Определить парадигму параллельного программирования для обработки результатов экспериментального исследования рис.1 с формированием результатов в виде таблицы, содержащей параметры обобщенного эксперимента

№ п.п.	Параметр	Обозначение
1	Математическое ожидание входного/выходного сигнала	MY_{inp}/ MY_{out}
2	Среднеквадратическое отклонение входного/выходного сигнала	SY_{inp}/ SY_{out}
3	Массивы локальных максимумов/минимумов	$maxY_i$/$minY_i$
4	Время начала переходного процесса, определяется превышением порога (вверх) $MY_{inp}+SY_{inp}$	t_1
5	Время завершения переходного процесса, определяется превышением порога (вниз) $MY_{out}+SY_{out}$	t_τ
6	Массив отметок времени локальных максимумов/минимумов	$t_{max\ i}$, $t_{min\ j}$

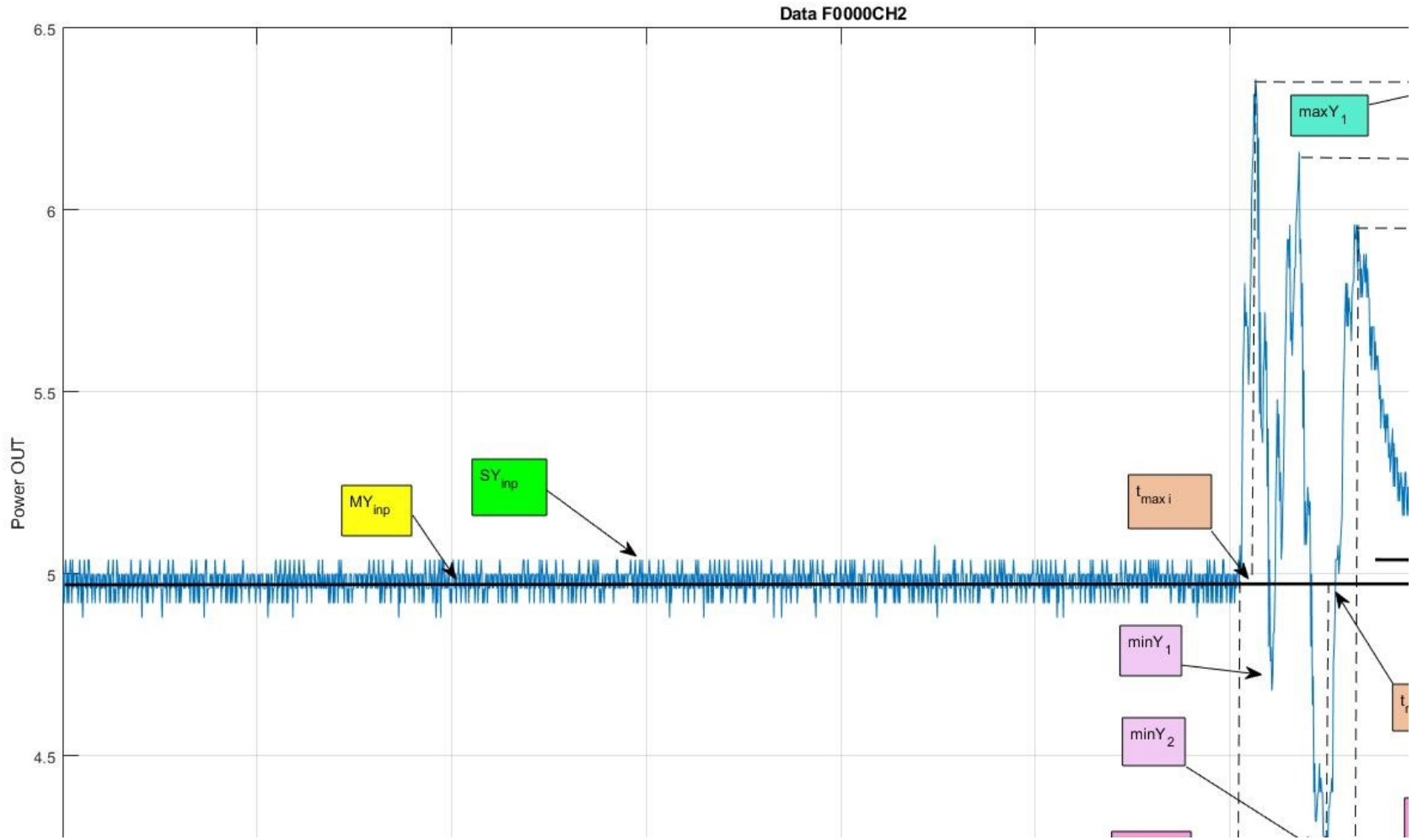

Рис.1. Параметры обобщенного эксперимента

5. Требования к оформлению отчета

Отчет должен содержать

1. Описание решения поставленной задачи в терминах выбранной парадигмы параллельного программирования
2. Алгоритм решения поставленной задачи
3. Структурная схема программного обеспечения
4. Программное обеспечение с опциями doxygen для связи с алгоритмом (C++ - 50%, Phyton – 50%)
5. Результаты (включающие имена исходных файлов):

- результаты преобразования исходных файлов;
- результирующие таблицы

Литература

1. https://hpc.icc.ru/foruser/library.html
2. Энтони Уильямс Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2012. – 672с.: ил.
3. ВВЕДЕНИЕ В ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ/ УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ В.В. Соснин, П.В. Балакшин, Д.С. Шилко, Д.А. Пушкарев, А.В. Мишенёв, П.В. Кустарев, А.А. Тропченко /ИТМО.-2023 https://books.ifmo.ru/file/pdf/3230.pdf
4. Способы реализации параллельных вычислений в программах на Python https://docs-python.ru/tutorial/mnogopotochnost-python/

Приложение 1. Терминология

1. ПАРАДИГМА (от греч. paradeigma – пример, образец).

- совокупность теоретических и методологических положений, принятых научным сообществом на известном этапе развития науки и используемых в качестве образца, модели, стандарта для научного исследования, интерпретации, оценки и систематизации научных данных, для осмысления гипотез и решения задач, возникающих в процессе научного познания.

2. CSV-файлы (файлы данных с разделителями-запятыми) — это файлы особого типа, которые можно создавать и редактировать в Excel. В CSV-файлах данные хранятся не в столбцах, а разделенные запятыми. Текст и числа, сохраненные в CSV-файле, можно легко переносить из одной программы в другую. Например, вы можете экспортировать контакты из Google в CSV-файл, а затем импортировать их в Outlook