Лекция 2

Средства разработки параллельных программ

Распараллеливание последовательных программ редко обеспечивает достижение приемлемого уровня параллелизма из-за ограничений вычислительного метода.

Трудно учесть особенности различных параллельных систем, так как чаще всего наиболее эффективные параллельные программы получаются, когда программа адаптирована под систему на которой реализуется.

* Программист может сам формировать последовательные фрагменты программы и следить за корректной синхронизации процессов
* Программист обязан чщаетльнос следить за распределения памяти и обязан соблюдать принцип единства присваивания
* При жесткой привязке к архитектуре, программист обязан обеспечить методику адаптацию программ к другой архитектуре
* С учетом архитектурных особенностей выполняется глубокая детализация распределения вычислений.

Создание параллельных программ использующих архитектуру смешанного типа, требует использование методов с явным описанем задачи, что усложняет.

В настоящее время существует 3 подхода к организации параллельных приложений:

* Многопоточные приложения. При этом программа содержит больше потоков чем процессорных элементов, способных их выполнить. Поэтому потоки выполняются поочереди. Примером является система реального времени, системы разделения времени, оконные приложения и тд. Основной проблеммой является борьба за разделяемый ресурс (память, процессор и тд)
* Распределенные системы. Класс программ, в которых различные участки кода выполняются на различных выч устройствах, связанных сетью. Основным способом является передача сообщений. Спецификой параллельных процессов является их относительная независимость и возможность обслучживания одним процессом различных клиентов. Подобное обслуживание проихожит по поступающим запросам. Примерами являются файловые серверы, системы СУБД, отказоустойчивые системы и тд
* Синхронизируемый параллельные вычисления. Цель: совместно быстро решить поставленную задачу. Такие приложения испоьльзуются в научных рассчетах, в моделировании и тд. Подобные программы требуют больших выч мощностей. Обычно число одновременно работающих процессов соотвествуют количеству процессоров.

Для эффективной разработки параллельных программ необходимо наличие программного инструментария. Разработчик должен эффективно испозовать средства, использование спец отладчиков, позволяющих разрабатывать программы для многопроцессорных систем.

* Для ВС с разделяемой памятью. Разработана под эгидой POSIX … содержит набор функции разработанных на С. Java – язык, который объединяет в себе много чего. openMP. Возможность преобразование последовательных программ в параллельную форму.
* Для систем с разделенной памятью.
* Объектно-декларируемые языки. Qt
* Параллельные-декларируемые языки.
* Итеративный параллелизм. Используется для реализации нескольких часто идентичных процессов, каждый из которых содержит циклы. Процессы работают совместно для решения общей задачи, взаимодействуют и синхронизируются с помощью разделяемых переменных или сообщений.
* Рекурсивный параллелизм. Используются в программах с одной или нескольких рекурсивными процедурами. Вызов должен быть независимый. Используется технология разделяй и властвуей
* Производители и потребители. Предполагает взаимодействие неравноправных процессов. Одни из которых производят данные, а другие потребляют. Часто образуют конвейер через который происходит обмен. В канал записывается связанная последовательность символов, прихводитель пишет в хвост, а потребитель считывает с головы канала. В общем случае канал или буфер имеют ограниченную длину, поэтому необходимо контролировать ситуацию, когда процесс производитель будет ожидать возможности записи в канал. При использовании систем с общей памятью, канал реализуется с помощью разделяемых перменных и примитивов синхронизации, таких как флаги и семафоры, в случае распределенных процессов для организации взаимодействия между производителем и потребителем используются send and receive. Организуется однонаправленный поток информации, производители и потребители являются независимыми процессами со своими потоками управления и скоростью работы. Взаимодействие идет только через канал.

Использование древовидной структуры. Реалзиуется при помощи метода сдваивания. Этот метод используется для распараллеливания алгоритмов связанных с обработкой различных массивов.

* Клиенты и серверы. Наиболее распространненая модель в распределенных системах. Клиентские процессы запрашивают у сервера некоторую информацию и ждут ответа, сервер ожитадет запрос и действует в сооствествии с ним. Сервер организован как одиночный, как и мноежственный. Может реализованная на распределенной системе, как и на одном компе. Сервер и клиент разные программы. Является типичной схемой писателей и читателей. Процесс-клиент запрашивает сервис и ждет обработки запроса, сервер ожидает запрос, обрабатывает и отсылает ответ, таким образом между процессами образуется двунаправленный поток информации. Каждый процессор реализуется алгоритмом. Запросы читателей обрабатываются независимо. Используется монопольный режим записи. Если клиент-сервер распределенная система, то доступ осуществляется при помощи передачи сообщения или вызов удаленных процедура.
* Управляющий и рабочие. Модель организации вычислений, при которой сущесвует некий поток, координирующий работу всех остальных потоков. Управляющий поток распределяет данные и собирает результаты. Чаще всего применяется на вычислительных системах с распределенной памятью. В этом случае любая переменная должна быть локальна для каждого процесса. Для обмена информации процессы будут реализовывать передачу сообщений. Для организации подобной работы удобно выделять управляющий процесс занимающийся распределением информации и собирающий результат работы.
* Взаимодействующие равные процессы. Данные модели не имеет управляющий поток, распределение работ осуществляется зараннее или динамически. Одним из распространненых способов является модель «портфель задач». Портфель – разделяемая переменная, которая чаще хранит число процессов для запуска. Все потоки равноправны. Портфель задач – счетчик процессов, подлежащих выполнению.

Для определения структуры программы необходимо определить понятия зависимых и независимых операций.

* Множество чтении операций. Множество переменных, которые в ходе операции читаются, но не изменяются
* Множество записей опеарции. Множество переменных, которые в ходе операции зименяются и возможно читаются.

Любые 2 операцции независимые, если их множество записей не пересекается. 2 операции могут выполнятся параллельно, если они независимы. Для определеня параллельного выпонения нескольких потоков или процессов используют следующую нотацию.

Co – содержит группы операторов, которые могут выполняться как независимые ветки.

Будет выполнятся значение цикла для каждого счетчика. Все операторы будут выполнятся только после того как будет законченно тело co секции. Неявная рабьерная синхронизация

Thread. Все что за секцией считается главным потоком, т.е. он сразу выполняется параллельно с итерациями циклов

Иттеративнй параллелизм обадает массоым параллелизмом. И может выполнено быть как в виде отдельных со процессов, так и в виде потоков. Если число процессоров в системе не превышает размерность матрицы, то распределение потоков по процессорам будет зависет от среды выполнения и реализации

Рекурсивный. Любую программу можно выполнять как множество параллельных процессов, если она содержит несколько неазвисимых рукурсивных вызова. Два вызова независимы, если их множество записей не пересекается. Это условие выполняется:

* Рекурсивная процедура не обращается к глобальным переменным по записи, а только читает
* Аргументы передаваемые в процедуру и результируещие переменные – разные переменные системы.

OPENMP – интерфейс для преобразования последовательной программы в параллельную для систем с общей памятью. Набор API функций ориентированных на написание многопоточных приложений. Разрабатывался для написания программ на Fortran, затем был перенесен на С.

Согласно OPENMP программа представлена как совокупность последовательных и параллельных областей. В момент запуска программы пророждается основной мастер-поток, который начинает выполнение со стартовой точки. Обычно мастер-поток организует последовательную область, затем реализуется вилочный параллелизм. Мастер-поток порождает набор потоков, а затем путем барьерной синхронизации собирает результат их работы перехода в последовательную область.

Все порожденные потоки, включая главный поток используют один и тот же код параллельной области. В параллельную область может быть вложена еще параллельная область и тд.

Создание параллельных программ состоит во внесении директив и функции в последовательную программу.

* Директивы. Pragma
  + Определение параллельной области. Parallel – создает область для следующего структурного блока. Эта директива сообщает компилятору, что этот блок должен выполняться параллельно в несколько потоков. Количество порождаемых потоков хранится в переменной среды. Если потоков больше, то происходит дублирование. В конце параллельной области обеспечивается барьерная синхронизация потоков. Дальнейшие вычисления будет делать основной поток. В данной директиве могут использоваться разделы:
    - If. Если истина, то код блока выполняется параллельно
    - Private. Содержит список переменных. Перечисляются переменные, которые локальные в потоках. Они инициируются в каждом потоке.
    - firstPrivate. Локальные переменные потоков, но инициализация идет в главном потоке
    - default. Определяет как воспринимать переменные внутри блока. Как общие или дополнительные определение
    - copyin. Значение переменных копируются всем нитям, но каждый поток будет работать со своей копией
    - reduction. Операторы и список переменных над которыми выполняются действия.

По умолчанию все переменные общие, кроме:

* + - Индексы параллельных циклов for локальные.
    - Переменные объявленные в параллельных областях локальные
    - Переменные private локальные
  + Разделение работы
  + Синхронизации

Разделы определяют модификаторы влияющие на поведение директивы.

* Функции