УДК 004

ОБЗОР МЕТОДОВ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Крыль Е.В., Хмелевой С.В.
Донецкий национальный технический университет
кафедра автоматизированных систем управления
E-mail: himikjeka@gmail.com

Аннотация

Крыль Е.В., Хмелевой С.В. Обзор методов параллельных вычислений. *Рассмотрена классификация архитектур вычислительных систем. Рассмотрены средства параллелизации ОС Linux и ОС Windows. Проанализированы Hardware и Software решения распараллеливания. Определены средства для распараллеливания решения задачи particle image velocimetry (PIV).*

Общая постановка проблемы

В наше время круг задач, требующих для своего решения применения мощных вычислительных ресурсов, постоянно расширяется. Это связано с тем, что произошли фундаментальные изменения в самой организации научных исследований. Вследствие широкого внедрения вычислительной техники, значительно усилилось направление численного моделирования и численного эксперимента. При этом численный эксперимент позволил значительно удешевить процесс научного и технологического поиска. Стало возможным моделировать в реальном времени процессы интенсивных физико-химических и ядерных реакций, глобальные атмосферные процессы, процессы экономического и промышленного развития регионов и т.д. Очевидно, что решение таких масштабных задач требует значительных вычислительных ресурсов. Самым перспективным из способов убыстрения решения задач выделяют параллелизацию алгоритмов.

Классификация архитектур параллельных вычислительных систем

Самой ранней и наиболее известной является классификация архитектур вычислительных систем, предложенная в 1966 году М.Флинном. Классификация базируется на понятии потока, под которым понимается последовательность элементов, команд или данных, обрабатываемая процессором. На основе числа потоков команд и потоков данных Флинн выделяет четыре класса архитектур: SISD,MISD,SIMD,MIMD.

SISD (single instruction stream / single data stream) - одиночный поток команд и одиночный поток данных (рис 1.а). К этому классу относятся классические последовательные машины, или иначе, машины фон-неймановского типа, например, PDP-11 или VAX 11/780. В таких машинах есть только один поток команд, все команды обрабатываются последовательно друг за другом и каждая команда инициирует одну операцию с одним потоком данных. Не имеет значения тот факт, что для увеличения скорости обработки команд и скорости выполнения арифметических операций может применяться конвейерная обработка - как машина CDC 6600 со скалярными функциональными устройствами, так и CDC 7600 с конвейерными попадают в этот класс.

SIMD (single instruction stream / multiple data stream) - одиночный поток команд и множественный поток данных (рис. 1.б). В архитектурах подобного рода сохраняется один поток команд, включающий, в отличие от предыдущего класса, векторные команды. Это позволяет выполнять одну арифметическую операцию сразу над многими данными - элементами вектора. Способ выполнения векторных операций не оговаривается, поэтому обработка элементов вектора может производится либо процессорной матрицей, как в

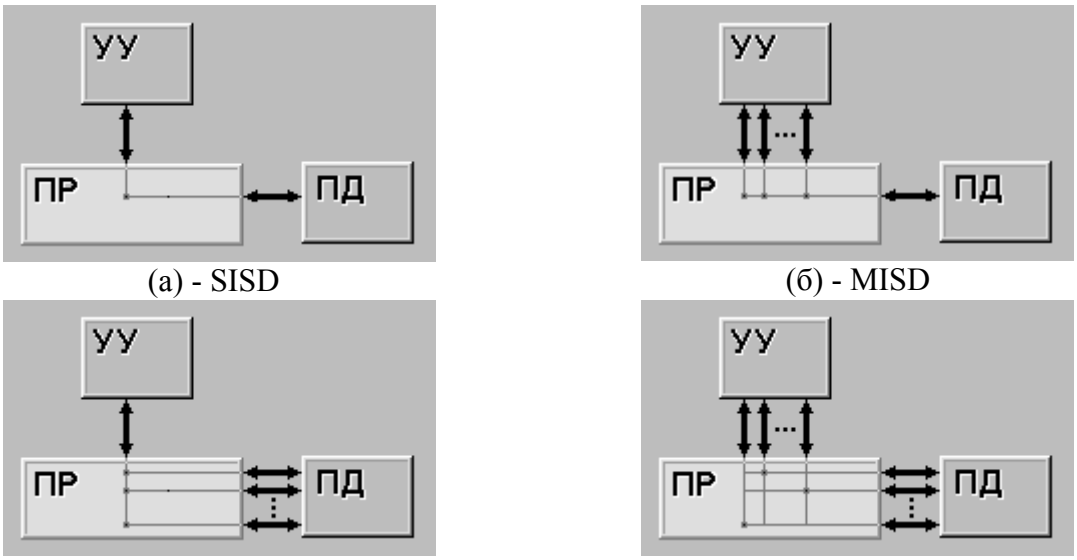
ILLIAC IV, либо с помощью конвейера, как, например, в машине CRAY-1.

MISD (multiple instruction stream / single data stream) - тип архитектуры параллельных вычислений, где несколько функциональных модулей (два или более) выполняют различные операции над одними данными (рис. 1.в).

Отказоустойчивые компьютеры, выполняющие одни и те же команды избыточно с целью обнаружения ошибок, как следует из определения, принадлежат к этому типу. К этому типу иногда относят конвейерную архитектуру, но не все с этим согласны, так как данные будут различаться после обработки на каждой стадии в конвейере. Некоторые относят систолический массив процессоров к архитектуре MISD.

Было создано немного ЭВМ с MISD-архитектурой, поскольку MIMD и SIMD чаще всего являются более подходящими для общих методик параллельных данных. Они обеспечивают лучшее масштабирование и использование вычислительных ресурсов, чем архитектура MISD [9].

MIMD (multiple instruction stream / multiple data stream) - множественный поток команд и множественный поток данных (рис. 1.г). Этот класс предполагает, что в вычислительной системе есть несколько устройств обработки команд, объединенных в единый комплекс и работающих каждое со своим потоком команд и данных [8].



(а) - SISD (б) - MISD

(в) - SIMD (г) - MIMD

Рисунок 1 - Виды архитектур по Флинну

Hardware – решения

Мощности современных процессоров достаточно для решения элементарных шагов большинства задач, а объединение нескольких десятков таких процессоров позволяет быстро и эффективно решать многие поставленные задачи, не прибегая к помощи мэйнфреймов и супер компьютеров.

Появилась идея создания параллельных вычислительных систем из общедоступных компьютеров на базе процессоров Intel и недорогих Ethernet-сетей, установив на эти компьютеры Linux и, объединив с помощью одной из бесплатно распространяемых коммуникационных библиотек (PVM или MPI) эти компьютеры в кластер. Эксперименты показали, что на многих классах задач и при достаточном числе узлов такие системы дают производительность, которую можно получить, используя дорогие суперкомпьютеры [1].

Также помимо кластерной технологии используется архитектура CUDA. CUDA – это программно-аппаратная архитектура, позволяющая производить вычисления с использованием графических процессоров NVIDIA, поддерживающих технологию GPGPU

(произвольных вычислений на видеокартах).

CUDA SDK позволяет программистам реализовывать на специальном упрощённом диалекте языка программирования Си алгоритмы, выполнимые на графических процессорах NVIDIA, и включать специальные функции в текст программы на Си. CUDA даёт разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического ускорителя и управлять его памятью, организовывать на нём сложные параллельные вычисления [2].

Также для решения задач параллельного программирования можно использовать многоядерные процессоры совместно со специальным ПО и библиотеками, например, Microsoft Parallel Extensions to the .Net Framework**.**

Средства ОС Linux

В ОС Linux можно выделить следующие локальные и удаленные средства межпроцессного взаимодействия:

- процессы;
- сигналы;
- трубы;
- блокировка файлов;
- очереди сообщений;
- семафоры;
- разделяемые сегменты памяти;
- потоки;
- сокеты;
- удаленный вызов процедур.

Рассмотрим подробнее некоторые из них.

Сигналы являются программными прерываниями, которые посылаются процессу, когда случается некоторое событие. Сигналы могут возникать синхронно с ошибкой в приложении, например SIGFPE (ошибка вычислений с плавающей запятой) и SIGSEGV (ошибка адресации), но большинство сигналов является асинхронными. Сигналы могут посылаться процессу, если система обнаруживает программное событие, например, когда пользователь дает команду прервать или остановить выполнение, или получен сигнал на завершение от другого процесса.

Труба является однонаправленным коммуникационным каналом между двумя процессами и может использоваться для поддержки коммуникаций и контроля информационного потока между двумя процессами. Труба может принимать только определенный объем данных (обычно 4 Кб). Если труба заполнена, процесс останавливается до тех пор, пока хотя бы один байт из этой трубы не будет прочитан и не появится свободное место, чтобы снова заполнить ее данными. С другой стороны, если труба пуста, то читающий процесс останавливается до тех пор, пока пишущий процесс не внесёт данные в эту трубу.

Разделяемые сегменты памяти как средство межпроцессной связи позволяют процессам иметь общие области виртуальной памяти и, как следствие, разделять содержащуюся в них информацию. Единицей разделяемой памяти являются сегменты, свойства которых зависят от аппаратных особенностей управления памятью.

Разделение памяти обеспечивает наиболее быстрый обмен данными между процессами.

Сокеты обеспечивают двухстороннюю связь типа ``точка-точка'' между двумя процессами. Они являются основными компонентами межсистемной и межпроцессной связи. Каждый сокет представляет собой конечную точку связи, с которой может быть совмещено некоторое имя. Он имеет определенный тип, и один процесс или несколько, связанных с ним процессов. Сокеты также можно использовать, чтобы организовать связь

между процессами на различных системах. Адресное пространство сокетов между связанными системами называют доменом Интернета. Коммуникации домена Интернета используют стек протоколов TCP/IP. Типы сокетов определяют особенности связи, доступные приложению. Процессы взаимодействуют только через сокеты одного и того же типа [6].

Средства ОС Windows

Как и в Linux в Windows имеются такие средства распараллеливания программ как именованные каналы (named pipes, трубы), сокеты, удаленный вызов процедур (класс технологий, позволяющих компьютерным программам вызывать функции или процедуры в другом адресном пространстве, как правило, на удалённых компьютерах).

Кроме этого в Windows существует еще ряд технологий, которые рассмотрены ниже.

Windows API (англ. application programming interfaces) — общее наименование целого набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows корпорации «Майкрософт». При помощи WinAPI можно реализовать многопоточные приложения.

.NET Framework - программная платформа, разработанная корпорацией Microsoft. С самого начала Microsoft .NET Framework предоставляла мириады низкоуровневых средств для создания параллельных приложений, в том числе целое пространство имен, специально выделенное для этой области: System.Threading. При наличии примерно 50 типов в этом пространстве имен в базовых сборках .NET Framework 3.5 (включая такие типы, как Thread, ThreadPool, Timer, Monitor, ManualResetEvent, ReaderWriterLock и Interlocked). На данный момент последней версией является .NET Framework 4 [7].

Software – решения

Для реализации кластерной технологии существует несколько коммуникационных библиотек. Рассмотрим две из них: PVM и MPI.

PVM (Parallel Virtual Machine) представляет собой программный пакет, который позволяет объеденять Unix и / или Windows компьютеры через сеть, и таким образом смогут использоваться как один большой параллельный компьютер. Таким образом большие вычислительные задачи могут быть решены более экономично и эффективно, используя совокупность мощности и памяти многих компьютеров. PVM позволяет пользователям использовать свои существующие аппаратные средства для решения проблемы при минимальных затратах [3].

MPI расшифровывается как "Message passing interface" ("Интерфейс передачи сообщений"). MPI - это стандарт на программный инструментарий для обеспечения связи между отдельными процессами параллельной задачи. MPI предоставляет программисту единый механизм взаимодействия процессов внутри параллельно исполняемой задачи независимо от машинной архитектуры (однопроцессорные, многопроцессорные с общей или раздельной памятью), взаимного расположения процессов (на одном физическом процессоре или на разных) и API операционной системы [4].

Для реализации параллельных вычислений на GPU (CUDA) или CPU (многоядерные процессоры) можно использовать OpenCL либо OpenMP.

OpenCL (от англ. Open Computing Language — открытый язык вычислений) — фреймворк для написания компьютерных программ, связанных с параллельными вычислениями на различных графических (англ. GPU) и центральных процессорах (англ. CPU) [5].

OpenMP (Open Multi-Processing) — открытый стандарт для распараллеливания программ на языках Си, Си++ и Фортран. Описывает совокупность директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для

программирования многопоточных приложений на многопроцессорных системах с общей памятью.

Краткое описание задачи и выбор средств ее решения

Задачей магистерской работы является исследование способов распараллеливания вычислительных процессов и применение одного из способов для ускорения работы реализации метода Particle Image Velocimetry (PIV) [10].

Метод PIV - цифровая трассерная визуализация потоков или лазерная анемометрия изображения частиц, позволяет измерять векторные поля скорости в выбранном сечении потока газа или жидкости.

В настоящее время имеется несколько реализаций этого метода, однако, ни одна из них не поддерживает распределенных параллельных вычислений, что могло бы в разы увеличить быстродействие. За счет увеличения производительности, станет возможным более оперативное изменение параметров исследуемого объекта.

Этот метод получил широкое распространение в различных областях науки и техники. Наиболее популярное применение – это моделирование поведения кипящего слоя.

Для решения поставленной задачи будет использоваться кластер, состоящий из нескольких компьютеров. В качестве операционной системы будет использоваться Windows, а в качестве средства распараллеливания вычислений используем сокеты. Для реализации кластерной технологии применим коммуникационную библиотеку MPI.

Данные средства были выбраны исходя из относительно низкой стоимости, доступности и сравнительно легкой реализации проекта. Также за счет такой структуры возможно удобное конфигурирование системы и повышение производительности путем добавления компьютеров в кластер.

Список литературы

1. Сбитнев Ю.И. Параллельные вычисления. [Электронный ресурс]. – 1998-2011. – Режим доступа: http://cluster.linux-ekb.info. – Загл. с экрана.
2. CUDA. [Электронный ресурс]. – 2012. – Режим доступа: http://ru.wikipedia.org/wiki/CUDA. – Загл. с экрана.
3. Parallel Virtual Machine. [Электронный ресурс]. – 2011. – Режим доступа: http://www.portablecomponentsforall.com/edu/pvm-ru. – Загл. с экрана.
4. Сбитнев Ю.И. Кластеры. Практическое руководство. – Екатеринбург, 2009. – 45 с.
5. OpenCL. [Электронный ресурс]. – 2012. – Режим доступа: http://ru.wikipedia.org/wiki/OpenCL. – Загл. с экрана.
6. Р.Х.Садыхов. Средства параллельного программирования для ОС Linux. [Электронный ресурс]. – Минск, 2004. – Режим доступа: http://www.opennet.ru/docs/RUS/linux_parallel. – Загл. с экрана.
7. Стефан Тауб. Прошлое, настоящее и будущее распараллеливания .NET-приложений. [Электронный ресурс]. – 2011. – Режим доступа: http://www.oszone.net/16322/net. – Загл. с экрана.
8. Воеводин В.В. Классификация Флинна. [Электронный ресурс]. – 2011. – Режим доступа: http://www.parallel.ru/computers/taxonomy/flynn.html. – Загл. с экрана.
9. MISD. [Электронный ресурс]. – 2011. – Режим доступа: http://ru.wikipedia.org/wiki/MISD. – Загл. с экрана.
10. Метод Particle Image Velocimetry: основы систем цифровой трассерной визуализации. [Электронный ресурс]. – 2011. – Режим доступа: http://cameraiq.ru/faq/flow_diagnostics/PIV_method/PIV_basic. – Загл. с экрана.