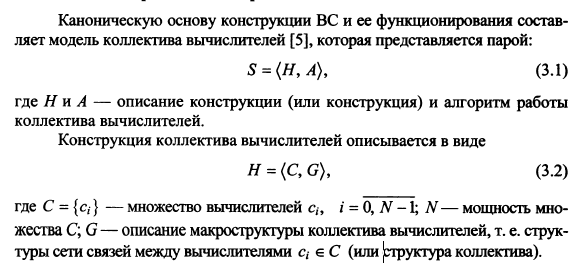
5. Архитектура вычислительных систем

5.1 Модель коллектива вычислителей. Структура ВС, алгоритм функционирования, модель ВС.



Конструкция коллектива вычислителей есть отражение следующих основополагающих архитектурных принципов:

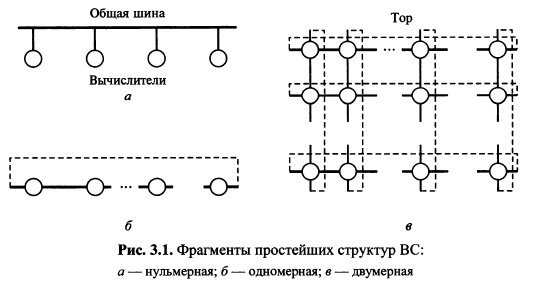
1) параллелизма (Parallelism, Concurrency) при обработке информации (параллельного выполнения операций на множестве C вычислителей, взаимодействующих через связи структуры G);

2) программируемости (Programmability, Adaptability) структуры (настраиваемости структуры G, достигаемой программными средствами);

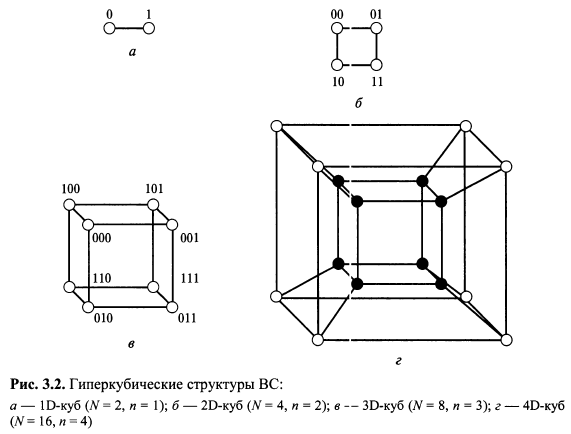
з) однородности (Нomogeneity) конструкции H (однородности вычислителей с C и структуры G).

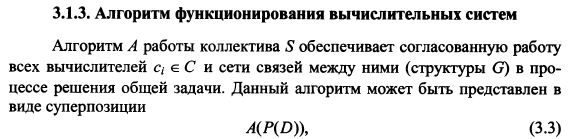
Структура (Structure, Topology) кoллeкmuвa вычислителей пpeдcmaвляemcя гpaфoм G, вершинам (узлам) кomopoгo сопоставлены вычucлumeлu ci ϵ C, a ребрам — линии связи между ними.

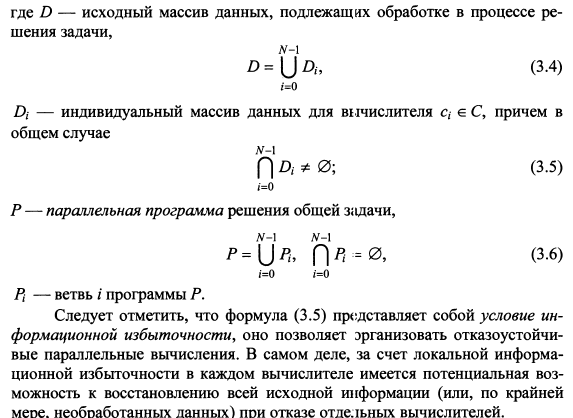
Простейшие структуры ВС. Различают нульмерные, одномерные и двумерные простейшие структуры ВС . B первом случае структура сети межвычислительньiх связей «вырождена», взаимодействие между вычислителями ВС осуществляется через общую шину (Common bus, Uni bus). В случае одномерньх структур (« инейки» Linear graph или «кольца» Ring) обеспечивается связь каждого вычислителя c двумя другими (соседними) вычислителями (рис. 3.1, a, б). B нульмерньх структурах имеется общий ресурс шина, в одномерньх же структурах этот ресурс трансформируется в распределенный, т. e. в локальные связи между вычислителями. Следовательно, архитектурные возможности (в частности, надежность) последних структyp существенно выше, чем y нульмерньх. Увеличение размерности структуры повышает структурную надежность ВС. B самом деле, двумерные структуры предоставляют каждому вычислителю непосредственную связь c четырьмя соседними. В качестве примеров двумерных структур (рис. 3.1, в) может служить «решетка» (точнее, 2D-решетка Two-dimensional griд) и тор (2D-тор Two-dimensional torus). Следовательно, в системах c двумерной структурой при отказах некоторых вычислителей и (или) связей между ними сохраняется возможность организации связных подмножеств исправных вычислителей.

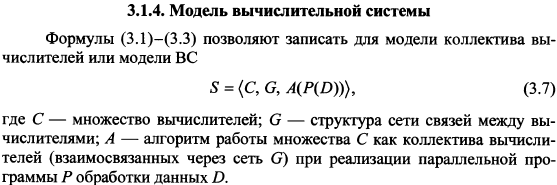


Гиперкубические структуры ВС. Гиперкубы, или структуры в виде булевых n-мepныx кубов, нашли широкое применение при построении современных высокопроизводительных ВС c массовым параллелизмом. Гиперкуб (Hypercube) по определению это однородный граф, для которого справедливо n = log 2 N, где N количество вершин; п число ребер, выходящих из каждой вершины; n называют также размерностью гип еркуба. Каждый вычислитель в гиперкубической ВС имеет связь ровно c n другими вычислителями.Гиперкуб размерности n называют также nD-кубом (D означает размерность (Dimension)).









Модель коллектива вычислителей допускает создание средств обработки информации разнообразных конфигураций. Представление коллектива вычислителей, в качестве макровычислителя делает возможным формирование сверхколлективов или систем коллективов, или макросы числительных систем. Модель коллектива вычислителей применима и на микроуровне; c ее помощью можно строить вы числитель как коллектив, составленный из микровычислителей.

5.2 Техническая реализация модели коллектива вычислителей и ее принципы. Архитектурные свойства ВС.

Каноническое описание модели коллектива вычислителей определяют принципы ее реализации. Выделим основные:

Модульность (Мodularity) принцип, предопределяющий формирование ВС из унифицированных элементов (называемых модулями), которые ф ункционально и конструктивно закончены, имеют средства сопряжения c другими элементами, разнообразие которых составляет полный набор. Функциональные и конструктивные возможности модулей, разнообразие их типов определяются исходя из требований, предъявляемыx к ВС, и, безусловно, из возможностей микpоэлектpонной базы.

Модyльность вычислительной системы обеспечивает:

1) возможность использования любого модуля заданного типа для выполнения любого соответствующего ему задания пользователя;

2) простоту замены одного модуля на другой однотипный;

з) масштабируемость, т. e. возможность увеличения или уменьшения модулей без коренной реконфигурации связей между остальными модулями;

4) открытость системы для модернизации, исключающую ее моральное старение.

Следует заметить, что принцип модyльности распространим и на средства программного обеспечения ВС.

При конструировании ВС c массовым параллелизмом достаточно ограничйться единственным модyлем-вычислителем, который бы обладал вычислительной и соединительной полнотой. Следовательно, модуль должен иметь средства автономного управления, располагать арифметико-логическим устройством и памятью и содержать локальный коммутатор - схемy для связи c другими модулями. На практике принято такой модуль-вычислитель называть либо элементарным процессором (ЭП), либо элементарной машиной (ЭМ). При этом считается, что ЭП это композиция из процессора и локального коммутатора.

Близкодействие (short-range interaction) - принцип построения ВС, обусловливающий такую организацию информационных взаимодействий между модyлями-вычислителями, при которой каждый из них может непосредственно (без «посредников») обмениваться информацией c весьма ограниченной частью модyлей-вычислителей. Следовательно, структура ВС позволяет осуществлять информационные взаимодействия между удаленными вершинами-вычислителями лишь c помощью промежуточных вершин-вычислителей, передающих информацию от точки к точке (point-to-point). Удаленными считаются те вершины в структуре ВС, расстояние между которыми более 1 (число ребер между которыми более 1). Принцип близкодействия допускает реализацию механизма управления ВС, не зависящий от количества составляющих ее вычислителей.

Вычислительные системы, основанные на принципах модульности и близкодействия, удовлетворяют также требованиям асинхронности, децентрализованности и распределенности.

Асинхронность функционирования (Asynchronous functioning) ВС обеспечивается, если порядок срабатывания ее модулей определяется не c помощью вырабатываемых тем или иным образе м отметок времени, a достижением заданных значений определенны (кап с правило, логических) функций. Использование асинхронных схем позволяет достичь в системе алгоритмически предельного быстpодействия: модули ВС срабатывают немедленно после выполнения соответствующего условия. Применение асинхронных схем обмена информацией между вычислителями позволяет не учитывать рaзброс в их тактовых частотах и колебания времени задержки сигналов в линиях связи.

Децентрализованность управления (Decentralized control) ВС достигается, если в системе нет выделенного модуля, который функционирует как единый для всей системы центр управления. децентрализованное управление системой основано на совместной работе всех исправных модулей системы, направленной на принятие решений, доставляющих оптимум выбранной целевой функции. Децентрализованное управление системой (в отличие от централизованного) позволяет:

1) достичь живучести ВС, т. e. ее способности продолжать работу при отказах модулей (в том числе и тек, которые предназначены для принятия решений);

2) избежать очередей при обслуживании заявок на управление.

Распределенность ресурсов (State of distribution) ВС позволяет создавать такую систему, в которой нет единого ресурса, используемого другими в режиме разделения времени. Под ресурсами ВС понимаются все объекты, которые запрашиваются, используются и освобождаются в ходе выполнения вычислений, например, процессоры или даже модули, входящие в их состав, модули оперативной памяти, внешние устройства, линии межмодульных связей, шины, файлы данных, компоненты ПО. Вместе c этим каждый ресурс распределенной ВС рассматривается как общий, доступный любому потребителю.

Архитектурные свойства ВС:

1. Масштабируемость (Scalability) ВС. Под масштабируемостью ВС понимается их способность к наращиванию и сокращению ресурсов, возможность варьирования производительности. Сложность (трудоемкость) задач, решаемых на вычислительных средствах, постоянно растет. Для сохранения в течение длительного времени способности ВС адекватно решать сложные задачи необходимо, чтобы она обладала архитектурным свойством масштабируемости. Это означает, в частности, что производительность, достигнутую ВС на заданном количестве вычислителей, можно увеличить, добавив еще один или несколько вычислителей.

2. Универсальность (Genericity, Generality, Versatility) ВС. Вычислительные системы алгоритмически и структурно универсальны. Принято считать, что ЭВМ (основанные на модели вычислителя) являются алгоритмически универсальными, если они обладают способностью (без изменения своих структур) реализовать алгоритм решения любой задачи. C другой стороны, ВС это коллектив вычислителей, каждый из которых обладает алгоритмической универсальностью, следовательно, и система универсальна (в общепринятом смысле).

3. Производительность (Performance, Throughput, Processing power) ВС. B отличие от ЭВМ, построенных на основе модели вычислителя, ВС не имеют принципиальных ограничений в повышении производительности. Увеличение производительности в них достигается з а счет не только повышения физического быстродействия микроэлектронных элементов, a главным образом увеличения числа вычислителей. При этом достигается простота настройки ПО на заданное число вычислителей в системе

4. Реконфигурируемость (Reconfigurability) ВС. Структурная и функциональная гибкости ВС обусловлены широкими возможностями систем по статической и динамической реконфигурации. Статическая реконфигурация ВС обеспечивается: варьированием числа вычислителей, их структуры и состава; выбором дня вычислителей числа полюсов для связи c другими вычислителями; возможностью построения структур в виде графов, относящиxся к различным классам; допустимостью применения в качестве связей каналов различных типов, различной физической природы и различной протяженности и т. п. Благодаря приспособленности ВС к статической реконфигурации достигается адаптация системы под область применения на этапе ее формирования.

Динамическая реконфигурация ВС поддерживается возможностью образования в системах таких (виртуальных) подсистем, структуры и функциональные организации которых адекватны входной мультипрограммной ситyации и структурам решаемых задач.

5. Надежность и живучесть (Reliability and Robustness) ВС.

Под надежностью ВС понимается ее способность к автоматической (программной) настройке и организации функционирования таких структурных схем, которые при отказах и восстановлении вычислителей обеспечивают заданный уровень производительности или, говоря иначе, возможность использовать фиксированное число исправных вычислителей (при реализации параллельных программ решения сложных задач).

Под живучестью ВС понимают свойство программной настройки и организации функционирования таких структурных схем, которые в условиях отказов и восстановления вычислителей гарантируют при выполнении параллельной программы производительность в заданных пределах или возможность использования всех исправных вычислителей.

6. Самоконтроль и самодиагностика (self-testing and self-diagnostics) ВС.

B системах-коллективах вычислителей может быть применен нетрадиционный подход к контролю и диагностике:

1) в качестве контpольно-диагностического ядра ВС могут быть использованы любые исправные вычислители и в пределе ядро любого произвольно выбранного вычислителя;

2) выбор ядра системы и определение ее исправности могут быть выполнены автоматически (c помощью средств ВС).

Предлагаемый подход позволяет говорить o самоконтроле и самодиагностике ВС.

7. Технико-экономическая эффективность (Technical-economical Efficiency) ВС.

5.3 Параллельные алгоритмы. Элементарные понятия параллельного программирования, параллельный алгоритм умножения матриц. Показатели эффективности параллельных алгоритмов. Парадокс параллелизма. Сложные задачи. Схемы обмена информацией между ветвями параллельной программы.

Параллельные алгоритмы

Описание процесса обработки информации, ориентированного на реализацию в коллективе вычислителей называют параллельным алгоритмом. Он в отличие от последовательного предусматривает реализацию на каждом временном шаге множества операций, и так же как и последовательный сохраняет зависимость последовательных этапов вычислений от результатов предыдущих. Запись параллельного алгоритма на языке, доступном коллективу вычислителей называют параллельной, а этот язык – параллельным языком программирования.

Методы вычислительной математики и соответствующие алгоритмы, как правило последовательный процесс приспособления алгоритмов к решению в коллективе вычислителей называют распараллеливанием.

Теоретическая и практическая деятельность, направленная на создание параллельных алгоритмов и программ называют параллельным программированием. Качество алгоритмов у такого программирования определяется методикой распараллеливания.

Существуют два подхода: Локальное и глобальное (крупноблочное) распараллеливание.

При первом подходе алгоритм решения задачи расщепляется на предельно простые блоки (операторы операция) и требуют максимального выделения этих операций для каждого шага вычислений. Такой подход считается не эффективным, т.к. он обеспечивает не однородную загрузку вычислителей. Его следует применять для оценки предельных возможностей по распараллеливанию. Практически эффективным считается глобальное распараллеливание. Оно требует расщепления вычислительного процесса на крупные блоки (подзадачи), между которыми существуют слабые связи. Последнее обеспечивает относительно незначительные расходы времени на реализацию обмена. Такие подзадачи называют ветвями параллельных программ.  
n-количество ветвей.

Основным приемом является распараллеливание по циклам. На практике такое распараллеливание по циклам, дает количество ветвей равное количеству повторений цикла. В параллельный программе всегда требуется выполнять обмены информацией между ветвями. Так же в этой программе имеются последовательные участки.

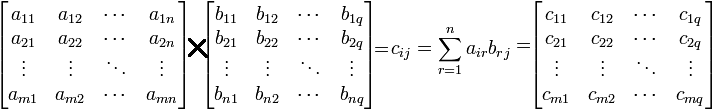
Например: в начале и в конце при вводе или выводе информации. В этой программе последовательные участки могут встречаться и внутри, они соответствуют негрупповым обменам информацией.

Параллельные алгоритмы умножения матриц.

Анализ прямых и итерационных методов в вычислительной математике показывает, что в них используется умножение векторов и матриц данных.

Выработаем алгоритм для умножения матриц большого размера:

A[1:m, 1:n]\*B[1:n, 1:q]=C[1:m, 1:q]

Пусть эту операцию умножения матриц нужно выполнить на N-вычислителях. Пусть m\*n и n\*q – числа, много больше количества вычислителей. (m\*n>>N, n\*q>>N)Это условие является необходимым. Распараллеливание следует применять для тех задач, которые недоступны для их решения на ЭВМ. Такие задачи называют сложными (трудоемкими). При параллельной обработке необходимо, чтобы каждый вычислитель рассчитывал N-ую часть элементов матрицы С. Следует заметить, что в матрицах А и В требуют размещения в каждом вычислителе n-ой части элементов соответствующей матрицы. Необходимо произвести однородное распределение элементов этих матриц по вычислителям.

Например, этого можно достичь, если матрицу A разделить на N горизонтальных полос, а матрицу В на N вертикальных полос. Каждая из этих полос будет находиться в своем вычислителе, т.е. в l-ом вычислителе будут размещены следующие матрицы А: (l-1)]m/N[+1,(l-1)]m/N[+2,…l]m/N[ B:(l-1)]q/N[+1,(l-1)]q/N[+2,…l]q/N[

-ближайшее к х целоечисло, такое, чтобы выполнялось неравенство: ]x[ >= x

Параллельный вычислительный процесс можно реализовать следующим образом.

Сначала первый вычислитель рассылает свою первую строку из матрицы А по всем остальным вычислителям. После этого все вычислитель параллельно рассчитывают свои элементы первой строки матрицы С. Затем первый вычислитель рассылает остальным вторую строку из своей полосы матрицы А и снова по тому же циклу. Рассылка элементов из первого вычислителя заканчивается после передачи всем другим строки]m/N[. Затем рассылкой занимается второй вычислитель и т.д. Применяя такое однородное распределение исходных данных и процедуру обработки мы получили параллельный алгоритм, состоящий из идентичных ветвей.



Показатель эффективности параллельных алгоритмов.

Ясно, что каждая ветвь параллельной программы это последовательность параллельных операторов (логических), а так же операторов, обеспечивающих работу между ветвями. Чем выше эффективность параллельного алгоритма, тем меньше взаимодействие.

В качестве одного из показателей эффективности можно использовать коэффициент накладных расходов ε=t/T, где t-время, расходуемое на организацию и реализацию обмена, Т-время, расходуемое на вычисление. Чтобы получить эффективный алгоритм, нужно, чтобы каждый вычислитель перерабатывал как можно больше данных.

Следующий показатель-коэффициент ускорения χ=- время решения этой же задачи на одном вычислителе (используется последовательная программа), -время решения этой же задачи на коллективе из n-вычислителей. Показатель χ показывает, насколько быстрее коллектив машин решает задачу быстрее, чем одна машина.

Коэффициент эффективности: Е=χ/N, N – количество ветвей программы.

Парадокс параллелизма:

χ>n, E>1 - состоит в достижении ускорения и эффективности, превышающих n и 1 соответственно. Парадокс выражается в более, чем линейном росте производительности параллельной ВС с увеличением количества n ее вычислителей.

Этот эффект постоянно шокирует специалистов, но парадоксом, по сути не является, этот эффект объясняется следующим:

1. Эффект памяти

2. Возможность применения алгоритмов параллельных методов решения задач, реализация которых невозможна на последовательных ЭВМ из-за ограничения емкости памяти.

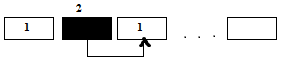
Пояснение к 1: Современная вычислительная система-коллектив вычислителей, каждый из которых имеет свою локальную память (ОЗУ). Сеть связей между вычислителями делает их локальную память общедоступной. Если суммарная локальная память всех вычислителей имеет достаточно большую емкость и допускает вложение параллельного алгоритма и данных. Следовательно, нет необходимости использовать более медленную внешнюю память, то будет достигнут минимум времени . С другой стороны, если сложность задачи достаточно высока и ее нельзя разместить в локальной памяти отдельного вычислителя и если она допускает решение на одном вычислителе только при использовании внешней памяти, то будет достигнуто значение , отличающееся от минимального и наконец, если время передачи между вычислителями сравнимо с временем обращения к локальной памяти, то будет иметь место следующее равенство: коэффициент , этим объясняется парадокс параллелизма.

Понятие сложной задачи.

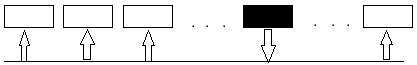
Ранее установлено, что эффективность реализации параллельных программ на вычислительных системах зависит от набора операций, которые следует выполнить при решении задачи, и от количества вычислителей, на котором реализуется параллельный алгоритм. В связи с этим, коэффициент накладных расходов можно написать в виде: где n-количество вычислителей, V-количество операций, которые следует выполнить при решении задач. n=const, при V->∞. Значение E будет практически удовлетворительным, если будет выполняться следующее: V=n\*k-число большее, либо равное 1 – эмпирический коэффициент основанный на практике, зависит от быстродействия каналов связи между вычислителями. Если скорость каналов сравнима со скоростью обращения к локальной памяти, то k=1. Если количество операций превышает количество вычислителей, хотя бы на порядок, то такую задачу лучше решать на параллельной системе. Задачу, удовлетворяющую количество операций, принято считать сложной. Следовательно, задача считается простой, если она может быть решена на одном вычислителе.

Схема обмена информацией между ветвями параллельного алгоритма.

Параллельный алгоритм-композиция ветвей, связанных для осуществления обмена. Возникает вопрос о многообразии схем обмена. Ясно, что для схемы может быть использована с помощью дифференцированного обмена. При этом обмене информация из одной ветви передается в другую.

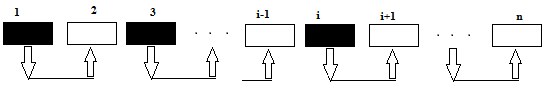
ДО: 

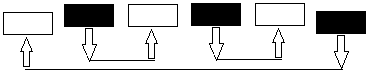
Каждая ветвь реализуется на своем вычислителе. Ясно, что при таком обмене участвуют 2 вычислителя, а все остальные простаивают: т.е. этот обмен неэффективен, соответственно возникает вопрос: существуют ли групповые или коллективные обмены, при которых все вычислители работают одновременно. Анализ параллельных алгоритмов решения сложных задач показывает, что к таковым обменам относятся трансляционные(ТО), трансляционно-циклические, конвейерно-параллельные обмены.

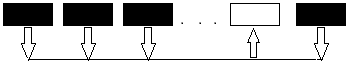
ТО (трансляционный)-из одного вычислителя информация передается всем остальным. Заметим, что данные схемы впервые были введены в 1960 году. Позднее они были обнаружены на Западе. В частности, трансляционный обмен получил название One-to-allBroadCast. 

Т-Ц (трансляционно-циклический) (all-to-allBroadCast)-суть обмена заключается в повторении n-раз трансляционного обмена из каждой машины.

К-П (ковейерно-параллельный)–реализуется в два или несколько этапов. Например обмен, состоящий из двух этапов. При четном n на первом этапе информация из вычислителей с нечетными номерами показывается в соседнем вычислителе с четными номерами. На втором этапе, из четных вычислителей передаются в соседние не четные вычислители, с большим номером.

1 этап

2 этап

В этом случае, все вычислители работают параллельно. Помимо этих алгоритмов обмена существует также коллекторный обмен который, по сути, является инвертированным трансляционным обменом, при котором информация из всех вычислителей собирается в одно. 

Данный метод тяжело реализовывать технически, потому его реализуют при помощи n-1 дифференциальных обменов. Следовательно, в коллекторном обмене в любой момент времени, работают только два вычислителя, остальные простаивают. Могут быть и другие типы обменов, но вышеизложенные – явно характерные.

Опыт применения крупноблочного распараллеливания сложных задач.

Опыт эксплуатации советских вычислительных систем с программной структурой полученный в Сибирском отделении Советской академии позволяет сделать следующие выводы:

1. Сложные задачи допускают представление в виде параллельного алгоритма с идентичными ветвями, значит, проблема программирования сводится к написанию программы для одной ветви.
2. При распределении исходных данных между ветвями параллельного алгоритма эффективным является принцип однородного расчленения массива.
3. Для построения параллельного алгоритма достаточно использовать пять схем обмена.
4. Простота схем обмена между ветвями позволяет ограничиться только однородными структурами, что обеспечивает высокую технико-экономическую эффективность ВС.
5. Для записи параллельного алгоритма решения сложных задач эффективны версии широко распространенных языков высокого уровня, которые называют параллельными языками. Они отличаются от последних тем, что в них включены операторы для организации взаимодействия между ветвями, например могут быть введены операторы, соответствующие схемам обмена. Дополнение к транслятору для параллельного языка оценивается в несколько %.
6. Простота схем обмена и распределения данных по ветвям ведет к простоте написания параллельных программ, при этом трудозатраты на разработку параллельных программ, в отличие от последовательных, не превышает 10%.
7. Схемы ТО ТЦО и КПО составляют более 90% от всех схем. Параллельные алгоритмы, построенные на их основе и методике крупноблочного распараллеливания, характеризуется несколькими процентами затрат времени на обмены по сравнению с общим временем решения задачи.

Некоторые замечания о MPI

Message Passing Interface – интерфейс передачи данных.

Этот инструментарий получил широкое распространение в 90-е г. Позволяет организовать схемы обмена между ветвями параллельной программы. Процедуры MPI совмещаются с той или иной схемой обмена информации. Соответственно, схемы обмена информацией и других систем взаимодействия поддерживались даже в первой в мире системе с программируемой структурой:1965-66 гг.

Минск-222. В ней была реализована библиотека с соответствующими процедурамиMPI-это подобие выполненного стандарта. Эти схемы являются важнейшими инструментами при разработке параллельных программ.

5.4 Концептуальное понятие о вычислительных системах. Типы архитектур: MISD, SIMD, MIMD.

Дать четкое определение понятию «система» нельзя. С греческого это целое, состоящее из чего-то. Система-множество элементов, находящихся в отношениях и связях друг с другом, которые образуют определенную целостность и единство. Под системой, в рамках данного предмета, понимают, средства обработки информации, основанные на модели коллектива вычислителей. При достаточно общей трактовке под вычислительной системой понимается совокупность взаимосвязанных и одновременно функционирующих аппаратурно-программных вычислителей ,которая способна не только реализовывать параллельный процесс решения сложной задачи, но и в процессе работы автоматически настраиваться и перестраиваться с целью достижения адекватности между своей структурно-функциональной организацией и структурой и характеристиками решаемой задачи.

Модель коллектива вычислителей есть диалектическая модель развития вычислителя, следовательно, вычислительная система существенно отличается от машины Фон-Неймана. Существует классификация Флинна (1965). Предложенную классификацию вычислителей по потокам команд и данным, имеющие место в вычислительном средстве.

Поток команд-любая их последовательность, подлежащая исполнению вычислительным средством. Для использования команд требуются данные, следовательно поток команд порождает поток данных.

Четыре типа архитектур по Флинну: SISD - ЭВМ; MISD, SIMD, MIMD–вычислительные системы.

Разнообразием отличается MIMD структура. Виды MIMD: мультипроцессорные, распределенные, с программируемой структурой.

Мультипроцессорные ВС представляют собой множество процессоров и некий единый ресурс – это общая память (как правило). Взаимодействие процессоров с памятью осуществляется через общую память.

Распределенные ВС: Все ресурсы представляются в виде композиции функционально-завершенных элементарных машин.

ВС с программируемой структурой: обладают возможностью автоматически подстраиваться под структуры и классы решаемых задач, т.е. в таких системах могут быть порождены виртуальные и специализированные ВС.

Кластерные ВС – введено компанией DEC. Кластер – группа компонентов, которые связаны между собой и функционирует как единое средство обработки информации. Это синоним ВС, но есть отличия. Кластер может иметь любую архитектуру, кроме SISD.

Общая трактовка: Вычислительный кластер – композиция множества вычислителей, сети связи между ними и ПО, предназначенное для реализации параллельных алгоритмов. Кластерные ВС могут строиться из специально разработанных компонентов, но главным образом, они строятся из серийно произведенных аппаратурно-программных средств. Последнее является принципом проектирования кластерных ВС.

Примеры:

SIMD – ILLIAC IV, CRAY-1

MISD – пуст, нет представителей

MIMD – C.mmp, Cm\*, Intel Paragon, CRAY Y-MP

