УДК 681.142.2

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ОДНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Ю.Г.Косарев, Н.Н.Миренков

Создание математического обеспечения общего назначения для современных вычислительных систем включает в себя решение двух основных проблем: разработку средств программирования и операционных систем. Создание средств программирования однородных вычислительных систем (ОВС) [1-3] связано:

1) с представлением процесса решения одной задачи в параллельном виде;

2) с предъявлением более высоких требований к качеству программирования по сравнению с программированием для последовательных ЭВМ.

Эти требования вытекают из самой природы решаемых на ОВС задач. Полнота использования технических возможностей вычислительной системы и качество программ влияют не только на себестоимость счета, но и на принципиальную возможность решения. Качество этих программ зависит от того, насколько хорошо решены следующие основные проблемы:

- равномерность и полнота загрузки процессоров;
- полнота использования памятей;
- рациональность организации работы с внешними устройствами;
- рациональное управление процессом счета, при котором доля затрат на управление значительно снижается;
- высокое качество собственно программирования и трансляции.

Специфика создания операционных систем связана с управлением параллельными процессами, структурой связей в ОВС и потоками задач, требующих для своей реализации объединенных ресурсов подсистем.

Сложность управления проявляется в функциональной неоднородности ОВС при одновременном решении на системе нескольких задач, что приводит к необходимости объединения в единое целое произвольно расположенных в системе элементарных машин (ЭМ).На функциональную неоднородность влияют различные приоритеты задач, а также выход ЭМ из строя.

Цель данной работы - рассмотреть основные пути решения указанных проблем.

I. Распараллеливание вычислительного процесса

Для решения задач на ОВС необходимо разделить вычислительный процесс на параллельные ветви, каждая из которых может выполняться автономным вычислительным устройством.Такое разделение возможно, в принципе, на разных уровнях. Рассмотрим это подробнее.

Схемы программ представляют собой иерархические структуры, высший уровень которых - крупные блоки,следующий уровень - подблоки крупных блоков и т.д.,самый низший уровень - операторы машинных команд.

Имеется два подхода при выборе уровней распараллеливания. Согласно первому - пооператорному [4-5], распараллеливание проводится на самом низком уровне. Согласно второму - крупноблочному [6-10], процесс распараллеливания начинается с самого высокого уровня. Переход на следующий уровень происходит только в том случае, если не удалось достичь нужной степени эффективного распараллеливания на данном.

Когда не удается удовлетворить требованиям эффективного распараллеливания ни на одном из верхних уровней,крупноблочный подход совпадает с пооператорным. Он позволяет находить нужные решения, как правило, без использования многочисленных и сложных связей, свойственных нижним уровням схем программ, и тем самым решать задачу распараллеливания в более простых условиях.

Разбиение программ на крупные блоки тесно связано с циклами. Поэтому наиболее частой формой крупноблочного распараллеливания является распараллеливание по циклам [6] (см.п. 4).

В зависимости от того,когда выполняется разделение вычислительного процесса на ветви и назначение ветвей на вычислительные устройства - заранее или во время счета,- распараллеливание подразделяется на динамическое и статическое.

Динамическое распараллеливание удобно при изменении ресурсов, отводимых данной задаче. Однако оно учитывает только локальные свойства программ и поэтому дает, как правило, неоптимальные решения. Кроме того,для принятия решений оно требует определенных затрат, значительно увеличивающих время счета.

Статическое распараллеливание свободно от указанных недостатков, однако требует дополнительных мер при изменении ресурсов, отводимых задаче.

По-видимому, наиболее рациональным способом распараллеливания является комбинированный, представляющий собой статическое распараллеливание, дополненное средствами подстройки параллельных программ к имеющимся ресурсам.

Следует отметить,что статические методы требуют наложения определенных ограничений на типы используемых программ.Однако, как показано в [2,6-10], эти ограничения почти всегда удовлетворяются на практике. Более того, для всех рассмотренных задач удалось построить универсальные программы, параметрически настраивающиеся на число машин как на параметр.

2. Языки

Параллельные алгоритмы для ОВС существенно отличаются от последовательных одновременностью выполнения большого числа операций. Формально отличие состоит в том, что вместо одномерной записи схемы алгоритма надо применять двухкоординатную, где одна координата показывает последовательность выполнения операций во времени, а вторая - распределение операций между ветвями вычислений.

Первым языком такого типа для ОВС был матричный р-язык [11]. С помощью этого языка можно задавать как алгоритмы ветвей (последовательных программ), так и взаимодействие между ни-

ми. Практическая реализация этой идеи впервые была осуществлена для системы "Минск-222" [12]. Языки АЛГОЛ, АКИ, ЛЯПАС были расширены специальными процедурами, позволяющими пользователям задавать взаимодействие ветвей на уровне команд системы (настройки, приема, передачи, синхронизации, обобщенного условного и безусловного переходов). Показано [13], что любой язык программирования расширением подобными процедурами можно превратить в язык для записи параллельных алгоритмов.

Опыт написания параллельных программ показал, что вместо процедур, реализующих команды системы, можно ввести макрооператоры, выполняющие основные схемы обменов и обобщенные условные переходы. Эти макрооператоры в большинстве случаев, когда нет особых требований к достижению максимальной производительности, устранили необходимость знания пользователями команд ОВС и её структуры, обеспечив вместе с тем достаточно высокую эффективность реализаций [14-15]. В этом плане проблему автоматизации написания параллельной программы по параллельному алгоритму можно считать решенной в той же степени, что и для современных ЭВМ, и достаточно удобной для практического использования. Рассмотренный подход сохраняет преемственность основных методов последовательного программирования и позволяет использовать существующие языки и трансляторы. Эволюция средств программирования в сторону этого подхода отмечается и в зарубежных работах, связанных с системами ILLIAC-IV [16] и [17].

В настоящее время имеется тенденция к некоторому упрощению структуры схем программ и запрещению использования меток и операторов типа GOTO, способствующих появлению значительного числа ошибок при написании программ [18].

К аналогичному запрещению указанных средств в языках пришли Ю.Г.Косарев и Р.М.Нуриев при исследовании проблем автоматического распараллеливания и преобразования алгоритмов. Эти средства оказались основной помехой для выявления циклов и приводили к алгоритмически неразрешимым проблемам.

3. Трансляторы

Построение языков параллельного программирования для ОВС путем расширения известных последовательных языков операторами системных взаимодействий позволяет для автоматизации программирования использовать существующие трансляторы. Этот подход был реализован для систем "Минск-222" и "Минимакс" [12,15], и его можно рассматривать как первый этап при создании средств параллельного программирования.

Получение высококачественных параллельных программ, удовлетворяющих требованиям, сформулированным во введении, – весьма сложная задача. Поэтому одним из реальных путей является построение системных трансляторов, представляющих собой последовательность открытых к изменениям версий, обладающих простотой включения по мере разработки средств, обеспечивающих выработку более эффективных программ. Принципы построения таких трансляторов, сформулированы в [19,20].

1. Принцип синтаксического управления и выделения языковой модели важен прежде всего тем, что он позволяет в процессе трансляции оперировать с синтаксическими конструкциями, имеющими законченное семантическое описание. Это создает основу для выделения независимых модулей транслятора и позволяет вести процесс трансляции в терминах, вытекающих из природы входного языка.

2. Принцип использования выделенного метаязыка обеспечивает запись грамматик входных языков в компактном и удобном для трансляции виде и допускает быстрые (безвозвратные) схемы синтаксического анализа и трансляции.

3. Принцип дедуктивного синтаксического анализа и итеративной структуры транслятора обеспечивает представление процесса трансляции в виде последовательно работающих блоков, каждый из которых управляется своей грамматикой и единой для всех блоков программой анализа.

4. Принцип привнесения семантики в синтаксические определения грамматик входных языков позволяет, с

одной стороны, иметь в трансляторе информацию о синтаксических конструкциях грамматик входных языков и их семантических эквивалентах, а с другой – вести процесс трансляции в соответствии с синтаксической структурой языка, разбивая его на последовательность законченных и естественных, вытекающих из природы языка, действий.

5. Принцип локализации изменений по конструкциям грамматик входных языков позволяет уменьшать чувствительность транслятора к изменению способов кодирования и трансляции исходной программы, системы ее редактирования; облегчает обнаружение и устранение ошибок или неудачных решений в трансляторе в процессе его отладки.

6. Принцип распараллеливания по циклам обеспечивает выбор удобного аппарата для преобразования последовательных циклов в параллельные. Особенностью такого распараллеливания является возможность при сравнительной простоте механизма распараллеливания получать универсальные программы, настраивающиеся на число машин в системе как на параметр.

На пути к созданию таких трансляторов в [21] разработан метаязык TR-грамматик (модификация R-грамматик [19]), отличительной особенностью правил которого является присутствие в них в явном виде элементов нетерминального словаря. Это позволяет в компактном и удобном для трансляции виде задавать грамматики входных языков и дает возможность с помощью нетерминальных символов транслировать синтаксические конструкции с законченным синтаксическим описанием. При этом легко выделяются независимые блоки транслятора.

Исследованы [22,23] вопросы структурной классификации TR-грамматик, выделены классы детерминированных и мононетерминальных грамматик, обладающих беступиковыми схемами синтаксического анализа и трансляции.

Подобно способам заданий языков с помощью автоматных преобразователей, в TR-метаязыке используется магазинная память. Однако в отличие, например, от автоматов с магазинной памятью, в TR-метаязыке: 1) правила грамматики с одинаковыми именами (аналог состояния магазинного автомата) сгруппированы в некоторые комплексы, каждому из которых присвоено имя, вынесенное за пределы соответствующих ему правил; 2) комплексы правил объединены в пакеты (атомные TR-грамматик), каждому из которых также присвоено имя, множество таких имен образует нетерминальный алфавит; 3) в правилах грамматик можно использовать как основные (терминальные) символы языка, так и нетерминальные.

Интерпретация правил TR-грамматики предполагает использование магазинной памяти с определенными над ней операциями записи, чтения и тождественной (пустой) операцией, причем операция чтения из магазина выполнима только тогда, когда он не пуст, остальные операции выполнимы всегда.

Доказан ряд теорем, устанавливающих совпадение классов языков, задаваемых ординарными и обобщенными TR-грамматиками, соответственно с классом языков, задаваемых ε-свободными КС-грамматиками, и классом рекурсивно-перечислимых множеств. Разработаны алгоритм построения TR-грамматик по грамматикам в нормальной форме Бэкуса и методы машинной реализации TR-грамматик.

Методика построения открытых к изменениям трансляторов для ОВС нашла свое воплощение при разработке транслятора ЭКСТРАН с языка высокого уровня [44, 45] для системы "Минск-222".

Дальнейшее развитие указанного направления – это включение в трансляторы блоков автоматического распараллеливания и преобразования программ, использующих в максимальной мере технические возможности ОВС.

4. Преобразования алгоритмов

Качество программы в значительной степени определяется тем, в какой мере она использует технические возможности вычислительного средства, т.е. обладает способностью приспосабливаться к особенностям техники, и в первую очередь, к числу процессоров и к объему памяти.

Соответственно могут быть введены два типа преобразований алгоритмов: пространственно-временные (ПВ) и информационно-временные (ИВ) [24]. Цель этих преобразований – получение последовательностей эквивалентных алгоритмов, которые перекрывают весь диапазон практических значений числа ветвей и объемов

информации. При этом, как показано в [24], естественно, накладываются ограничения на степень близости качества алгоритмов, что позволяет для каждой задачи ограничить количество алгоритмов, подлежащих рассмотрению, сравнительно небольшим числом.

ПВ- и ИВ-преобразования алгоритмов тесно связаны со свойствами циклов. На этой основе удалось найти условия, при которых ПВ-преобразования могут выполняться путем изменения определенных параметров.

Есть основания надеяться, что и для ИВ-преобразований также удастся найти удобную форму реализации.

Проблема алгоритмической разрешимости при выполнении преобразований алгоритмов и построении эффективных р-программ требует некоторой дополнительной информации о свойствах алгоритмов. Предполагается [25] эту информацию локализовать в синтаксической форме программ, названных циклическими программами, в которых все повторяющиеся процессы представлены в виде циклов.

Цикл состоит из тела – подпрограммы, также являющейся циклической программой, и оператора, проверяющего некоторые условия повторной выполнимости тела. Для циклов, использующих массивы, требуется, чтобы тело имело вид: $A_1^n A_1^{n-1} \dots A_1^1 A_2$, где A_1^i и A_2 – подпрограммы, а индексы переменных, употребляемых в A_1^i и A_2, являлись выходными переменными хотя бы одной из предыдущих подпрограмм A_1^{i+k} для A_1^i и A_1^i для $A_2(i,k,l \geq 1)$. A_1^i называется о р г а н и з а т о р о м i-ого уровня, A_2 – ядром цикла, а сам цикл – р а з д е л е н н ы м.

Разделение тела цикла на ядро и организаторы связано с тем, что для конечности числа повторений цикла необходимо, чтобы каждое следующее повторение использовало результат предыдущего повторения. То есть в теле должны осуществляться два различных действия: изменение индексов переменных и выполнение после этого операторов. Поэтому в соответствии с методом максимального синтаксиса (см. [25]) они должны быть синтаксически выделены.

Циклические программы характеризуются следующим:

1) Время выполнения и объем используемой памяти почти полностью определяются циклами. Поэтому для настройки этих программ на ресурсы системы необходимо преобразование только циклов.

2) Класс циклических программ функционально полон.

3) В циклических программах запрещается использовать некоторые конструкции, затрудняющие отладку программ.

4) Рекурсивные и стандартные схемы эффективно сводимы к схемам циклических программ.

Доказан ряд теорем, устанавливающих необходимые и достаточные условия распараллеливания основных конструкций в циклической программе: цикла и пары циклов, один из которых вложен в другой. Установлена полнота этих двух способов распараллеливания для построения параллельных программ, в которых одновременно могут выполняться только повторения тел, организаторов и ядер циклов исходной программы. Показана (см. [2]) возможность эффективного построения универсальной программы, реализующей циклические программы на многопроцессорной системе с любым заданным числом процессоров. Установлено, что язык, допускающий только циклические программы с разделенными циклами, является языком высокого уровня для написания программ многопроцессорных систем.

Теоретические исследования по циклическим программам дополнены специальной программой, проверяющей условия, при которых справедливы основные теоремы.

5. Анализ задач

Опыт применения крупноблочного распараллеливания для большого круга задач [9] позволил сделать следующие выводы об особенностях взаимодействия между ветвями параллельного алгоритма.

1. Время обмена информацией между ветвями параллельного алгоритма, отнесенное к общему времени счета, для большинства задач есть величина порядка $1/cn$ (где 1 – число ветвей, n – параметр задачи, характеризующий объем вычислений, c – константа, учитывающая различие затрат времени на обменные и на вычислительные операции).

2. Все многообразие схем обмена сводится к пяти типам передачи информации:

– трансляционный обмен – из одной ветви во все остальные;

– трансляционно-циклический обмен – трансляционный обмен в цикле по всем ветвям;

– обмен сдвигом – из каждой ветви в соседнюю с бо́льшим (меньшим) номером;

- дифференцированный обмен – из одной ветви в некоторые выделенные;

- коллекторный обмен – из всех ветвей в одну выделенную.

Подавляющее большинство схем обмена приходится на первые три. На них же падает почти все время, затрачиваемое на обмены. Поэтому эти три схемы и определяют требования к технической реализации системы коммутации в ОВС.

3. При распределении исходных данных между ветвями параллельного процесса эффективным является принцип однородного распределения массивов [10], при котором:

- объемы распределяемых частей массивов равны;

- нумерация распределяемых частей массивов соответствует нумерации ветвей;

- массивы разделяются на части параллельными прямыми (многомерные массивы – параллельными плоскостями);

- массивы или их части дублируются в ветвях одинаковым образом.

Применение этого принципа приводит к следующим основным способам разбиения массивов: полосами (по отношению к одной координате), циклическими полосами, полосами с дублированием смежных элементов, скошенными полосами (одновременное распределение по двум координатам).

4. Простота схем обмена и распределения данных обусловливает простоту реализации параллельных программ. Благодаря этому, например, при решении задач на системе "Минск-222" дополнительные затраты на организацию системных взаимодействий составляют, как правило, менее 10% от общего объема программы.

5. Важное свойство распараллеливания по циклам – хорошая сочетаемость его с методами эффективного использования суммарного объема оперативных памятей системы, т.е. возможность проведения как пространственно-временных, так и информационно-временных преобразований алгоритмов [9], что при решении многих задач на системе дает существенный выигрыш во времени.

Таким образом, анализ задач подтвердил основные положения методики крупноблочного распараллеливания: возможно эффективное разбиение вычислительного процесса на большое число ветвей уже на верхних уровнях структурной иерархии программ и построение универсальных программ, параметрически настраиваемых на число машин в системе.

6. Отладка программ

Процесс создания высококачественных параллельных программ, как и любых программ с большой вычислительной сложностью, включает в себя решение двух основных задач: выявление и устранение ошибок (собственно отладка) и совершенствование программы (оптимизация). При параллельном программировании имеет смысл решать обе указанные задачи совместно [26,27].

Собственно отладка параллельных программ сводится к отладке участков, не содержащих системных взаимодействий, и проверке правильности задания взаимодействий между различными ветвями вычислений. Для действий первого типа можно использовать известные отладочные средства, для действий второго – необходимы дополнительные блоки.

Поскольку совокупность допустимых ситуаций, имеющих место при взаимодействии ветвей, представляется в виде множества предложений некоторого языка, то для распознавания ситуации можно эффективно использовать методы синтаксического контроля [28].

Обнаружение семантических ошибок может быть осуществлено путем введения в процедуры, соответствующие системным макрооператорам [15], специальных средств, которые во время реализации параллельных программ проводят анализ взаимодействия ветвей и, кроме того, обеспечивают выдачу обобщенной информации об аварийных завершениях в ветвях и защиту операционной системы от неопределенных (тупиковых) ситуаций.

Оптимизационные действия связаны с выдачей информации, используя которую можно совершенствовать параллельную программу и алгоритм [15].

К такой информации относятся:

- время простоев машины, знание которого позволяет по-другому распределять данные между ветвями;

- время выполнения блока программы, важное для реализации счета в случаях конвейерных схем и при неидентичных ветвях;

- точность вычислений, облегчающая обнаружение функций, на которых имеют место наибольшие потери точности;

- объем оперативной памяти и объем вычислений над ней, позволяющие судить об активности использования памяти разными блоками программы.

Комплекс отладочных и анализирующих средств для выполнения рассмотренных здесь действий реализован для ОВС "Минск-222" [26-27] и разрабатывается для системы "Минимакс" [15].

7. Средства специальной подготовки программ

Для повышения эффективности функционирования ОВС могут быть применены некоторые специальные программные системы-утилиты для сегментирования ветвей параллельных программ и их предварительной подготовки на случай продолжения счета при выходе машин системы из строя.

1. Использование методики распараллеливания по циклам обычно приводит к параллельным программам в виде совокупности идентичных ветвей. При этом одновременное выполнение одинаковых ветвей не требует обязательного хранения соответствующей программы целиком в каждой машине. Обычно достаточно иметь одну программу, распределенную по системе, и представлять одновременно всем машинам только ту часть (сегмент), которая реализуется в данный момент.

Эффективность такого разбиения определяется выполнением некоторых условий, связанных с объемами сегментов, и с временными затратами на загрузку сегментов и синхронизацию машин [29-30].

Реализация этой задачи сводится к трем основным этапам:

- эффективное разбиение на сегменты, удовлетворяющие заданным условиям;
- настройка сегментов на специальное поле оперативной памяти;
- распределение сегментов по машинам и диспетчирование их загрузкой.

В [30] разработаны алгоритмы, которые могут быть положены в основу специального блока-сегментатора, выполняющего в режиме диалога с пользователем подготовку р-программы к "сегментированному" счету.

2. Цель предварительной подготовки параллельных программ - минимизировать замедление вычислений, связанное с а) тестированием аппаратуры ОВС в процессе счета, б) выполнением повторного счета, в) возобновлением вычислений после устранения неисправностей [31].

В процессе предварительной подготовки выясняются:

- способ требуемой реализации параллельных программ: счет с пошаговым тестированием аппаратуры, двойной или с полным дублированием на нескольких подсистемах;
- необходимость дополнительного пошагового запоминания содержимого памяти машин между двумя тестированиями аппаратуры или сравнениями результатов вычислений;
- точки в программе пользователя для вставки специальных операторов, инициирующих обращения к операционной системе для непосредственного запуска тестов или взвода таймера (здесь же осуществляется вставка этих операторов);
- составы тестов для проверки аппаратуры, используемой на каждом шаге счета;
- особенности программы и способ продолжения вычислений при обнаружении неисправности (перестройка программы для подсистемы с меньшим числом машин, переконфигурация системы и замена неисправной машины для подсистемы такого же размера);
- необходимость использования подпрограмм, осуществляющих дополнительный контроль обменных взаимодействий между машинами;
- фрагменты программы, выполняющие неповторимые вычисления (например, для процессов, реализуемых в реальном времени).

Средства предварительной подготовки программ должны дополняться специальными блоками операционной системы для диспетчирования загрузкой сегментов, запоминания содержимого регистров и памяти, запуска тестов, перестройки программ и переконфигурации подсистем.

8. Управляющие системы

Программное управление структурой ОВС обеспечивает достаточную универсальность применений системы как для решения сложных задач, требующих объединенных ресурсов, так и для потока мелких задач, взаимодействие между которыми может быть редким и неравномерным.

Для реализации этой универсальности, необходимой во многих крупных автоматизированных системах управления, предложены четыре режима работы [32,33], специализированные по методам управления и способам использования ресурсов поступающими заданиями: автономной работы, параллельной обработки, диспетчера и профилактики.

В режиме автономной работы пользователь непосредственно связан с одной из машин системы (а через нее и со всей системой).При этом обеспечивается возможность счета,трансляции, отладки,редактирования,сегментирования на одной машине, доступ к системной информации, формулирование заданий системе и т.п.

В режиме параллельной обработки реализуются параллельные программы и организуется продолжение счета при сбоях или выходе ЭМ из строя.

В режиме профилактики тестируются машины программными и аппаратными средствами.

В режиме диспетчера выполняются работы,связанные с управлением системой.

Для принятия решений в сложных мультипрограммных ситуациях, когда необходимо, с одной стороны, действовать немедленно, а с другой - попытаться лучше понять ситуацию, структура управляющей системы выбирается иерархической с блоками, названными "Главный диспетчер", "Старший диспетчер", "Диспетчер".

Г л а в н ы й диспетчер осуществляет разбиение системы на подсистемы, определяет режим работы для подсистем,управляет операциями ввода/вывода для системных внешних устройств и т.п.

При реализации последовательности таких разбиений "Главный диспетчер" организует динамическое функционирование ОВС при решении заданного набора программ или стохастическое функционирование при обслуживании потока программ [34].

Сложность работы с потоком заданий на параллельную обработку объясняется как случайным характером поступлений заданий, так и случайным временем их обслуживания.Задача состоит в том, чтобы определить методику оперативного, связанного с конкретной ситуацией выбора размера подсистемы, обслуживающей параллельную обработку,и назначения последовательности функциональных состояний для этой подсистемы. Предложена следующая стратегия. В каждый момент времени в системе обычно существует набор заданий, известны их ранги, предполагаемое время выполнения, приоритеты (ограничения на порядок выполнения), а также максимально возможное время реализации (при превышении которого задание снимается с обслуживания). Используя эти данные и алгоритмы из [35], выбирают размер временного кванта τ, в течение которого не предполагается менять функциональное состояние подсистемы. На основании полученного значения τ вычисляют-

ся размер подсистемы и последовательность (во времени) её разбиений на более мелкие подсистемы.

Поскольку со временем ситуация может измениться (например, появится новый набор заданий),то необходима коррекция обслуживания. Пересмотр расписания, определенного старым набором заданий, реализуется в следующих случаях:

- при отсутствии в старом наборе заданий для загрузки некоторых подсистем до завершения временного кванта, что связано с несоответствием предполагаемого и фактического времени решения задач;
- при наличии в новом наборе заданий с высоким приоритетом.

В функции "Главного диспетчера" также входит распознавание подсистем заданных структур [36] и выбор пространственного размещения для подсистем.Для каждого режима важно выявить наиболее часто используемые схемы обменов и, опираясь на это,найти наиболее эффективные структуры.

С т а р ш и й диспетчер управляет работой подсистем.Выделяют три типа "Старших диспетчеров": для автономной работы, режимов профилактики и параллельной обработки. Основные функции этих диспетчеров: реализация очередности обслуживания ЭМ (подсистем) на основе заявленного приоритета или вида выполняемой работы [37], организация доступа к системным данным, формирование заданий "Главному диспетчеру",тестирование ЭМ и т.п.

Д и с п е т ч е р управляет работой ЭМ: обеспечивает вход в систему и распознавание директив пользователя, взаимодействует со "Старшими диспетчерами",анализирует аварийные ситуации,участвует в реализации операторов обменных взаимодействий и др. Особенностью диспетчеров ЭМ является их способность организовать децентрализованное функционирование ОВС при автономном поступлении ветвей параллельных программ в разные машины. Основные алгоритмы планирования, которые используются блоками высших уровней иерархии, можно разбить на три группы:

- количественное распределение ЭМ по режимам работы на основе заданных для режимов приоритетов,поступивших заданий (и предполагаемых поступлений), эффективности загрузки подсистем и числа исправных ЭМ;

– количественное распределение ЭМ по подсистемам (в рамках режима параллельной обработки) на основе характеристик поступающих заданий: рангов, времени выполнения, приоритетов или других отношений следования;

– пространственное размещение подсистем на основе данных о профилактическом обслуживании ЭМ и координатах этих ЭМ в ОВС, а также информации о размещении и объемах файлов, требуемых для выполнения поступивших заданий.

Целевыми функциями задач, определяемых алгоритмами планирования, являются максимизация загрузки ресурсов системы и минимизация времени обменов между ЭМ.

Для уменьшения накладных расходов, связанных с работой управляющей системы, по-видимому, достаточно эффективным будет наличие нескольких для каждой задачи алгоритмов, среди которых имеются как универсальные для любых мультипрограммных ситуаций [34,38], так и специализированные [39–40] для наиболее характерных случаев. Тот или иной набор алгоритмов должен выбираться на уровне генерации управляющей системы.

Многогранность процесса генерации математического обеспечения для ОВС проявляется не только на уровне выбора алгоритмов планирования, но и на уровнях режимов работы и соответствующих им приоритетов, числа машин в ОВС и принципов доступа к системе, структуры ЭМ и состава внешних устройств. Причем выделяются два уровня генерации: 1) стратегический, на котором принимаются решения глобального характера на достаточно длительный срок, и 2) тактический, на котором в процессе функционирования системы принимаются решения локального характера, связанные с оперативной подстройкой блоков программного обеспечения к характеристикам поступающих заданий и режиму работы, в котором в данный момент каждая машина работает.

В результате работ, проведенных в Институте математики СО АН СССР и ряде других организаций нашей страны, созданы основы теории математического обеспечения для ОВС и разработаны пути его практической реализации.

Опыт создания систем "Минск-222", "Минимакс" и др. [12–15, 33, 41, 43] показывает, что разработка математического обеспечения для ОВС должна выполняться в неразрывной связи с созданием аппаратных средств (особенно с принятием решений о структуре связей в системе) и опираться на математическое обеспечение элементарных машин. Это позволяет за достаточно короткое время разрабатывать математическое обеспечение для конкретных систем и обеспечивать преемственность основных, привычных для программиста методов работы.

Литература

1. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во СО АН СССР, 1962.

2. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука", 1966.

3. ЕВРЕИНОВ Э.В., ХОРОШЕВСКИЙ В.Г. Однородные вычислительные системы. Настоящий сборник, стр. 32–60.

4. КОТОВ В.Е. Теория параллельного программирования. Прикладные аспекты. –"Кибернетика", 1974, № 1. с. 3–17.

5. SCHWARTZ E.S. An Automatik Sequencing Prosedure with Application to parallel Programming. – "J.Assac. Compt. Machinery", 1961, vol.8, N 4. p. 513–537.

6. КОСАРЕВ Ю.Г. Распараллеливание по циклам. –В кн.: Вычислительные системы. Вып. 24. Новосибирск, "Наука", 1967, с. 3–20.

7. КОСАРЕВ Ю.Г. О методике решения задач на универсальных вычислительных системах. –В кн.: Вычислительные системы. Вып.17. Новосибирск, 1965, с.61–99.

8. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. О решении задач на универсальных вычислительных системах. –Там же, с. 106–164.

9. КОСАРЕВ Ю.Г. О схемах обмена между ветвями параллельных алгоритмов. –В кн.: Вычислительные системы. Вып.51. Новосибирск, 1972, с. 70–75.

10. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на однородных вычислительных системах. –В кн.: Вычислительные системы. Вып. 57. Новосибирск, 1973, с. 3–32.

11. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Матричный р-язык для описания параллельных алгоритмов. –В кн.: Вычислительные системы. Вып.17. Новосибирск, 1965, с. 100–105.

12. ГОЛОВЯШКИНА Л.В., КОЛОСОВА Ю.И., КОСАРЕВ Ю.Г., МИРЕНКОВ Н.Н. Автоматизация программирования на основе существующих трансляторов. –В кн.: Вычислительные системы. Вып.30. Новосибирск, 1968, с.63–69.

13. КОСАРЕВ Ю.Г. Об автоматизации программирования для ОВС. –В кн.: Труды I Всесоюзной конференции по вычислительным системам. Вып.4. Новосибирск, "Наука", 1966, с. 75–79.

I4. ГРИШАЕВА Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И., КОНСТАНТИНОВ В.И., КОРНЕЕВ В.Д., МИРЕНКОВ Н.Н. Язык параллельных алгоритмов. -В кн.: Вычислительные системы. Вып.57. Новосибирск, I973, с. 33–54.

I5. КЕРБЕЛЬ В.Г.,КОЛОСОВА Ю.И.,КОРНЕЕВ В.Д.,МИРЕНКОВ Н.Н., ЩЕРБАКОВ Е.В. Средства программирования системы "МИНИМАКС".Отчет ИМ СО АН СССР, Новосибирск, I974.

I6. KUCK D.J. ILLIAC IV software and application programming. - "IEEE Trans.Computers",1968,vol.17,N 8,p.758–770.

17. WILSON D.E. The PEPE support software system.- COMPCON 72. 6th Ann.IEEE Comput.Soc.Int.Conf., San Francisco, Calif., 1972. Dig.Pap. 1972: Innov.Comput,Archit.N.Y.1972,p.61–64.

18. DONALDSON J.R. Structured Programming. - "Datamation", 1973, December, vol.19, N 12,p.50–52.

I9. ГЛУШКОВ В.М., ВЕЛЬБИЦКИЙ И.В., СТОГНИЙ А.А. Об одном подходе к построению системного математического обеспечения современных вычислительных машин. -"Кибернетика", I972, № 3, с. 25–35.

20. ВЕЛЬБИЦКИЙ И.В., КОСАРЕВ Ю.Г. Системный подход к построению трансляторов для вычислительных систем. -В кн.: Вычислительные системы. Вып. 42. Новосибирск, I970, с. I2–2I.

2I. КОНСТАНТИНОВ В.И., НУРИЕВ Р.М.Метаязык трансляции контекстно-свободных языков.-В кн.: Вычислительные системы.Вып.57. Новосибирск, I973, с. 74–83.

22. КОНСТАНТИНОВ В.И. Классы TR-грамматик. -В кн.: Вычислительные системы. Вып. 57. Новосибирск, I973, с. 84–90.

23. КОНСТАНТИНОВ В.И. Некоторые вопросы построения трансляторов для ОВС. Автореф. канд. дисс. Киев, I974 (ИК АН УССР).

24. КОСАРЕВ Ю.Г. О пространственно- и информационно-временных преобразованиях алгоритмов. -В кн.: Вычислительные системы. Вып. 57, Новосибирск, с. I49–I60.

25. КОСАРЕВ Ю.Г., НУРИЕВ Р.М.Программы с циклической структурой. Отчет ИМ СО АН СССР, Новосибирск, I974.

26. КОЛОСОВА Ю.И. Комплекс средств производства параллельных программ. -В кн.: Вычислительные системы. Вып. 57. Новосибирск, I973, с. 98–II4.

27. КОЛОСОВА Ю.И. Об отладочных программах для ОВС "Минск-222". -В кн.: Вычислительные системы. Вып. 5I. Новосибирск, I972, с. 76–8I.

28. КОЛОСОВА Ю.И., МИРЕНКОВ Н.Н. Исследование взаимодействий ветвей р-программ. -В кн.: Вычислительные системы. Вып.57. Новосибирск, I968, с. II5–I24.

29. ГАМИДОВ В.М., МИРЕНКОВ Н.Н. Задача сегментирования р-программ. -В кн.: "Вычислительная техника" (Тезисы научно-технической конференции), Новосибирск, I972, с. 34–36.

30. ГАМИДОВ В.М., МИРЕНКОВ Н.Н. Алгоритмы сегментирования параллельных программ. Отчет ИМ СО АН СССР, Новосибирск,I974.

3I. КОРНЕЕВ В.Д., МИРЕНКОВ Н.Н. Повышение надежности вычислений на основе предварительной подготовки параллельных программ. Отчет ИМ СО АН СССР, Новосибирск, I974.

32. МИРЕНКОВ Н.Н. Диспетчирование в однородных вычислительных системах. -В кн.: Материалы 3 Всесоюзной конференции по ОВС и средам. Таганрог, I972, с. 5I–54.

33. МИРЕНКОВ Н.Н. "МИНИМАКС" - вычислительная система коллективного пользования. Отчет ИМ СО АН СССР,Новосибирск,I974.

34. КРЫЛОВ Э.Г., МИРЕНКОВ Н.Н. Алгоритмы планирования функциональных состояний ОВС.Отчет ИМ СО АН СССР,Новосибирск,I974.

35. МИРЕНКОВ Н.Н.Алгоритмы планирования для диспетчера однородной вычислительной системы. -В кн.: Вычислительные системы. Вып. 42. Новосибирск, I970, с. 34–46.

36. МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Алгоритмы распознавания подсистем заданных структур в ОВС. Отчет ИМ СО АН СССР, Новосибирск, I974.

37. КЕРБЕЛЬ В.Г., КОРНЕЕВ В.Д.,МИРЕНКОВ Н.Н.,ЩЕРБАКОВ Е.С. Управляющая программа системы "Минимакс". Отчет ИМ СО АН СССР, Новосибирск. I974.

38. ХОРОШЕВСКИЙ В.Г. Об алгоритмах функционирования однородных универсальных вычислительных систем. -В кн.: Вычислительные системы. Вып. 39. Новосибирск, I970, с. 3–I4.

39. КОНСТАНТИНОВ В.И., МИРЕНКОВ Н.Н. Функционирование однородной вычислительной системы в режиме потока больших задач. -В кн.: Вычислительные системы.Вып.42.Новосибирск,I970,с.47–58.

40. ГОЛОСКОКОВА Т.М., ХОРОШЕВСКИЙ В.Г.Алгоритмы функционирования ОВС в простейших ситуациях. -В кн.: Вычислительные системы. Вып. 39. Новосибирск, I970, с. I5–29.

4I. ВИНОКУРОВ В.Г., ДИМИТРИЕВ Ю.К., ЕВРЕИНОВ Э.В., КОСТЕЛЯНСКИЙ В.М., ЛЕХНОВА Г.М., МИРЕНКОВ Н.Н., РЕЗАНОВ В.Е., ХОРОШЕВСКИЙ В.Г. Однородная вычислительная система из мини-машин. -В кн.: Вычислительные системы. Вып. 5I. Новосибирск, I972, с. I27–I46.

42. RILEY W.B. Minicomputer network - challenge to maxi - computer? - "Electronics",1971,N 29,p.56–62.

43. КОСАРЕВ Ю.Г., ЮЩЕНКО Ю.Л. Некоторые проблемы организации вычислительного процесса на системах. -В кн.: Труды семинара "Автоматизация программирования". Вып.I. Киев,I968,с.I–8.

44. КОНСТАНТИНОВ В.И. Транслятор для однородной вычислительной системы. -В кн.: Вычислительные системы. Вып.5I. Новосибирск, I972, с. 59–69.

45. ГРИШАЕВА Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И.,КОРНЕЕВ В.Д. КОНСТАНТИНОВ В.И., МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Транслятор с языка параллельных алгоритмов. -В кн.: Вычислительные системы. Вып.57. Новосибирск, I973, с. 55–73.

Поступила в ред.-изд.отд.
I7 мая I974 года