УДК 681.3.06:519

Н.С. КОВАЛЕНКО, В.Н. ВЕНГЕРОВ

ПРИЕМЫ УСКОРЕНИЯ ВЫЧИСЛЕНИЙ ПРИ РАСПРЕДЕЛЕННОЙ И ВЕКТОРНО-КОНВЕЙЕРНОЙ ОБРАБОТКАХ ДАННЫХ

Рассматриваются приемы ускорения вычислений при распределенной и векторно-конвейерной обработках данных нескольких функциональных устройств за счет избыточности данных и операций при условии низкой интенсивности потоков межпроцессорных передач данных.

1. Постановка задачи и основные понятия

Эффективное применение суперкомпьютеров с целью решения прикладных задач возможно лишь при удовлетворительном решении проблемы отображения алгоритмов на их архитектуры. Эта проблема, в свою очередь, порождает множество задач как технологии организации параллельных вычислений, так и разработки эффективных приемов ускорения вычислений. В статье проведена классификация приемов ускорения вычислений по признакам относительной производительности и обоснована эффективность в зависимости от числа операций программных реализаций и объемов обрабатываемых данных. Определены приемы ускорения вычислений, в основе которых лежит структурная избыточность данных и показывается их эффективность в программных реализациях алгоритмов, базирующихся на графовых структурах данных. Устанавливается зависимость времени выполнения программных реализаций от используемых структур данных. Показывается, что время выполнения программных реализаций алгоритмов с квадратичной трудоемкостью может быть значительно улучшено за счет изменения используемых структур данных.

Распределенные вычисления – перспективная и динамично развивающаяся область организации параллелизма [1]. Этот термин обычно используется при параллельной обработке данных нескольких функциональных устройств, достаточно удаленных одно от другого, в которых передача данных по линиям связи приводит к существенным временным задержкам. Эффективная обработка данных при таком способе организации вычислений возможна только для алгоритмов с низкой интенсивностью потоков межпроцессорных передач данных. Перечисленные условия характерны, например, при организации вычислений в многомашинных вычислительных комплексах (ВК), высокопроизводительных вычислительных кластерах, локальных или глобальных информационных сетях [2]. В связи с этим производительность при распределенной обработке параллельных процессов может существенно колебаться в зависимос-

ти от целого ряда факторов в достаточно широком диапазоне, а ускорение вычислений может быть достигнуто более богатым набором приемов.

Аппарат теории графов, являясь мощным математическим средством, получил широкое распространение при разработке, тестировании, анализе и оценке сложности, а также распараллеливании последовательных программ. Он позволяет наглядно представить совокупность операций алгоритма, связь между отдельными операциями и порядок их выполнения.

Представим множество операций, выполняемых в исследуемом алгоритме решения задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного мультиграфа $G = (V, R)$, называемого *графом алгоритма* (ГА) [1], где $V = \{1, 2, ..., |V|\}$ есть множество вершин графа, представляющее выполняемые операции алгоритма, а R – множество дуг графа, причем дуга $r = (i, j)$ принадлежит графу G, если операция j использует результат выполнения операции i. Предположим, что множество вершин V разбито на такие непересекающиеся подмножества $V_1, V_2, ..., V_k$, что если $u \in V_i$, $v \in V_j$ и существует дуга (u, v), то $i < j$. В этом случае разбиение $V = \bigcup_{i=1}^{k} Vi$ называется *строгой параллельной формой алгоритма*, а подмножества $V_1, V_2, ..., V_k$ – ее *ярусами* или *уровнями*. Существует строгая параллельная форма, при которой максимальная из длин путей, оканчивающихся в вершине с индексом i, равна $i - 1$. Для этой параллельной формы число используемых индексов на единицу больше критического пути графа. Строгая параллельная форма называется *канонической*, если все входные вершины находятся в группе с одним индексом, равным единице. Для заданного графа алгоритма его каноническая форма *единственна*. Число k называют *высотой* параллельной формы, число вершин в ярусе ($|Vi|$, $i = \overline{i,k}$) – *шириной* i-го яруса, а максимальную ширину ярусов ($\max|Vi|$, $i = \overline{i,k}$) – *шириной* параллельной формы [1].

Необходимо отметить, что множество V может быть разбито на два непересекающихся подмножества V_c и V_B скалярных и векторных операций соответственно. Тогда *трудоемкостью*, или *числом операций*, алгоритма будет называться число

$$N = |Vc| + \sum_{i \in V_B} Li ,$$

где $|V_c|$ – число элементов множества V_c, а L_i – длина вектора, используемого в i-й векторной операции [3]. Заметим, что в общем случае в оценку числа операций алгоритма необходимо включать также и число условных и безусловных переходов.

Пусть задан произвольный ГА и две его программные реализации P_1 и P_2 соответственно последовательного и распределенного на p процессорах алгоритмов решения задачи, для которых определены величины N_i и T_i – число операций и время выполнения соответствующих программных реализаций, $i = 1, 2$.

Известно [3], что производительность программной реализации P_i вычисляется по формуле

$$\nu_i = \frac{N_i}{T_i}, \; i = 1, 2.$$

Ускорением вычислений будем называть отношение времени выполнения программной реализации P_1 ко времени выполнения программной реализации P_2: $\alpha = T_1/T_2$. Все преобразования графа алгоритма или соответствующей ему программной реализации, приводящие к ускорению вычислений без нарушения их функциональных возможностей при получении конечного результата, будем называть *приемами ускорения* вычислений. Приемы, для которых $\alpha > 1$, будем называть *эффективными.*

Эффективность использования p процессоров при решении задачи распределенным алгоритмом определяет среднюю долю времени выполнения алгоритма, в течение которой процессоры реально используются для решения задачи, и определяется соотношением $\beta = \alpha/p$.

Величину $\psi = N_2/N_1$ будем называть *относительной трудоемкостью*, а величину $\varphi = \nu_2/\nu_1 = \alpha\psi$ – *относительной производительностью* программной реализации P_2 по отношению к P_1.

При $\psi < 1$ величину $V_\psi = 1/\psi$ будем называть *выигрышем от уменьшения трудоемкости*, а при $\psi > 1$ величину $P_\psi = 1/\psi$ – *проигрышем в трудоемкости.* Аналогично, при $\varphi > 1$ величину $V_\varphi = \varphi$ будем называть *выигрышем в производительности*, а при $\varphi < 1$ величину $P_\varphi = 1/\varphi$ – *проигрышем в производительности* программной реализации P_2 по отношению к P_1.

С учетом введенных обозначений ускорение вычислений α можно представить в следующем виде: $\alpha = \varphi/\psi$.

2. Приемы ускорения вычислений и их классификация

Для ЭВМ последовательного типа одним из наиболее эффективных приемов ускорения вычислений является снижение числа операций в программных реализациях и самих алгоритмах. При распределенной обработке параллельных процессов производительность многопроцессорной системы может существенно колебаться в зависимости от целого ряда факторов в достаточно широком диапазоне. Следовательно, ускорения вычислений можно достичь более богатым набором приемов по сравнению с ЭВМ последовательного типа.

Утверждение 1. *Величина ускорения вычислений при распределенной обработке будет эффективной* ($\alpha > 1$), *если выполняется хотя бы одно из следующих условий:*

1) $\varphi < 1$, $\psi < 1$ *и* $\psi < \varphi$, *т.е. уменьшаются относительная производительность системы распределенной обработки и число операций в программной реализации, причем число операций должно сократиться в большее число раз, чем уменьшится производительность;*

2) $\varphi = 1$, $\psi < 1$, *т.е. относительная производительность системы распределенной обработки не изменяется, а сокращается число операций в программной реализации;*

3) $\varphi > 1$, $\psi < 1$, *т.е. относительная производительность распределенной обработки повышается и сокращается число операций в программной реализации;*

4) $\varphi > 1$, $\psi = 1$, *т.е. повышается относительная производительность системы распределенной обработки, а число операций не изменяется;*

5) $\varphi > 1$, $\psi > 1$ *и* $\psi < \varphi$, *т.е. повышается относительная производительность системы распределенной обработки и увеличивается число операций в программной реализации, причем относительная производительность должна повыситься быстрее, чем увеличится число операций.*

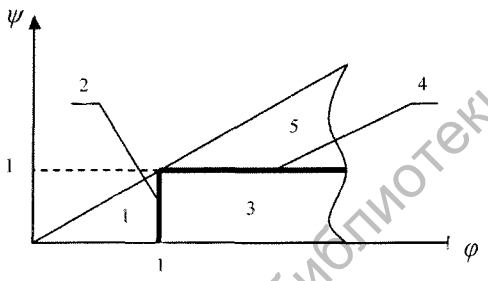

Рис. 1. Приемы ускорения вычислений при распределенной обработке

Данное утверждение позволяет классифицировать приемы ускорения вычислений по признаку относительной производительности распределенной обработки на пять групп, которые схематически изображены на рис. 1, где первой группе соответствует нижняя треугольная область, второй – вертикальная полоса, третьей – прямоугольная область, четвертой – горизонтальная полоса, а пятой – верхняя треугольная область.

Из утверждения 1 следует, что для систем распределенной обработки приемы ускорения вычислений применительно к числу операций программной реализации, или алгоритма, могут быть отнесены к одному из трех следующих классов.

1) Приемы ускорения вычислений, основанные на уменьшении числа операций программной реализации или самого алгоритма. Производительность систем распределенной обработки при этом может уменьшаться, оставаться постоянной или увеличиваться. К данному классу относятся приемы групп 1–3.

2) Приемы ускорения вычислений, для которых число операций программной реализации и алгоритма не изменяется. При этом повышается производительность системы распределенной обработки. К данному классу относятся приемы группы 4.

3) Приемы ускорения вычислений, основанные на избыточности операций или данных. При этом повышается производительность системы распределенной обработки наряду с увеличением числа операций или

объема обрабатываемой информации. К данному классу относятся приемы группы 5.

3. Эффективность приемов ускорения вычислений

Вполне естественным является вопрос об эффективности той или иной группы приемов ускорения вычислений. Но однозначного вывода сделать нельзя, поскольку следует учитывать особенности конкретного алгоритма и решаемой задачи. Несмотря на это, можно оценить потенциальные возможности рассмотренных групп приемов ускорения вычислений, характеризующих следующее утверждение.

Утверждение 2. *Группа приемов ускорения вычислений при распределенной обработке будет эффективной* ($\alpha > 1$), *если* α *удовлетворяет условиям:*

1) $\alpha = V_\psi / p_\varphi$, *т.е. ускорение вычислений равно отношению выигрыша от уменьшения трудоемкости программной реализации к проигрышу производительности системы распределенной обработки;*

2) $\alpha = V_\psi$, *т.е. ускорение вычислений равно выигрышу от уменьшения трудоемкости программной реализации;*

3) $\alpha = V_\psi V_\varphi$, *т.е. ускорение вычислений равно произведению выигрыша от уменьшения трудоемкости программной реализации и выигрыша производительности распределенной системы;*

4) $\alpha = V_\varphi$, *т.е. ускорение вычислений равно выигрышу производительности распределенной системы;*

5) $\alpha = V_\varphi / p_\psi$, *т.е. ускорение вычислений равно отношению выигрыша производительности распределенной системы к проигрышу в трудоемкости программной реализации.*

Как следует из утверждения 2, наибольшими потенциальными возможностями ускорения вычислений обладают приемы третьей группы.

4. Приемы ускорения вычислений за счет избыточности операций и данных при векторно-конвейерной обработке

Предположим, что при модификации некоторой программной реализации увеличивается число ее операций или, другими словами, увеличивается число вершин ГА. Добавляемые в ГА вершины будем называть *избыточными*, а все остальные – *необходимыми*. Если избыточные вершины исключают использование некоторых других вершин (при этом подразумевается, что число добавляемых вершин больше числа исключаемых), то такие вершины назовем *заменяемыми избыточными*, в противном случае – *добавляемыми избыточными*. Эти термины в одинаковой мере могут относиться к данным, операциям и функциональным устройствам.

Множество вершин V в ГА разбивается на два непересекающихся подмножества: вершины критического пути (МВКП) и фоновые вершины (МФВ).

К приемам ускорения вычислений за счет добавляемой избыточности могут быть отнесены следующие:

– искусственная задержка выдачи команд, т.е. добавление новых операций в МФВ для организации режима зацепления по двум операндам при работе с векторными регистрами;

– устранение входных зависимостей и антизависимостей по данным в циклических участках программ путем введения промежуточных переменных или массивов. В этом случае увеличивается число элементов в МВКП;

– расщепление циклического участка на несколько циклов с целью достижения его частичной векторизации. При этом происходит увеличение эпилогов циклов и увеличение числа условных переходов, т.е. увеличивается число элементов МВКП.

К приемам ускорения вычислений за счет заменяемой избыточности относятся следующие:

– замена последовательной работы одного функционального устройства параллельной работой или работой в режиме зацепления с другими функциональными устройствами;

– векторизация циклов с условными операторами. По условному выражению формируется двоичная векторная маска (управляющий вектор), которая позволяет затем выбрать элементы векторов для вычислений или записи в оперативную память;

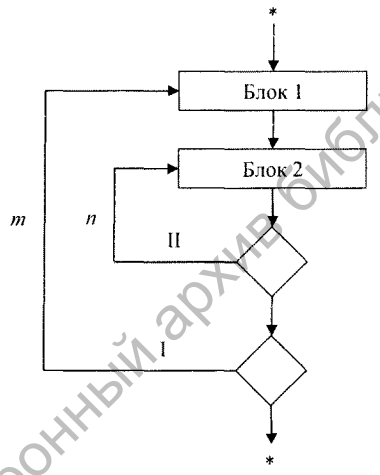

Рис. 2. Блок-схема циклического участка программы

– внесение операций в тело цикла. Данный прием рассмотрим на следующем примере. Пусть задан циклический участок программы, блок-схема которого приведена на рис. 2, где α_1 и α_2 – числа операций, выполняемых соответственно в блоках 1 и 2, а β_1 и β_2 – времена реализации этих блоков. Цикл I выполняется m раз, а цикл II – n раз. Без учета команд условных переходов число операций этого участка составляет величину $N = (\alpha_2 n + \alpha_1)\, m$, а время его реализации будет $T = (\beta_2 n + \beta_1)\, m$. Рассмотрим следующий обмен между МВКП и МФВ блоков 1 и 2. Операции из МВКП блока 1 переносятся в МФВ блока 2. В этом случае число операций увеличивается и составляет величину $N^* = ((\alpha_2 + \delta)\, n + + \alpha_1 - \delta)\, m$, где δ – число переносимых операций, а время выполнения программной реализации уменьшается и составляет величину $T^* = ((\beta_2 n + \beta_1) - \beta)\, m$, где β – время реализации переносимых операций в МВКП блока I;

– структурная избыточность данных. Данный прием ускорения вычислений состоит в замене нерегулярной структуры данных на регулярную с целью исключения косвенной адресации и достижения максимальной производительности ВК ЭВМ. Влияние избыточности данных на

ускорение вычислений рассмотрим на примере комбинаторных задач анализа топологии графов, для решения которых необходима полная информация обо всех дугах и вершинах.

Одним из способов представления графа является представление с помощью списков инцидентности, которое требует $m + n$ ячеек памяти, где m и n – число дуг и число вершин графа соответственно. Другим способом представления графа является матрица смежностей, которая требует n^2 ячеек памяти. Разность между этими объемами ($n^2 - m - n$) будем называть *структурной избыточностью данных*. Однако, несмотря на структурную избыточность данных, этот способ позволяет эффективно применять векторную обработку и за счет этого достигать максимальной производительности векторно-конвейерных ЭВМ, что невозможно при использовании списков инцидентности из-за косвенной адресации данных. Кроме того, с учетом булевости матрицы смежностей графа и архитектуры ВК ЭВМ объем избыточной информации сокращается до $n[n/64] - m - n$ машинных слов ($[x]$ – наименьшее целое большее или равное x) за счет побитового представления информации.

Возможности ускорения вычислений на ВК ЭВМ за счет избыточности данных продемонстрируем на примере программной реализации алгоритма поиска в глубину [4]. Рассмотрим две программные реализации данного алгоритма на языке Ассемблер ВК ЭВМ, основные положения которого изложены в [3].

1) Реализация на структуре данных, заданной списками инцидентности в режиме скалярной обработки ВК ЭВМ.

В этом случае время работы программы составляет $T_1 = 36m + 85n + 250$ тактов ВК ЭВМ, а число операций $N_1 = 14m + 30n + 50$, где m и n – число дуг и число вершин графа соответственно. При этом производительность ВК ЭВМ приближается к полувекторной ($v_1 \approx 0{,}40$).

2) Реализация на структуре данных, заданной матрицей смежностей при ее побитовом представлении в режиме векторной обработки ВК ЭВМ.

При данном способе реализации

$$T_2 = \begin{cases} (96 + 2k)n + 250 & \text{тактов при } 5 \le k \le 12; \\ (73 + 4k)n + 250 & \text{тактов при } 13 \le k \le 63. \end{cases}$$

Здесь $k = \max\{5, [n/64]\}$, а число операций $N_2 = (47+7k)n + 50$. Производительность ВК ЭВМ приближается к векторной и супервекторной ($v_2 \in [0{,}75;1{,}5]$), т.е. в 2–4 раза выше.

Приведенные выше оценки получены из анализа соответствующих программных реализаций. На основе таких оценок можно установить зависимость времени реализации алгоритмов от объема избыточной информации для данных программных реализаций.

Утверждение 3. *Если число вершин графа менее 768 ($1 < n < 768$), то при любом объеме избыточной информации при реализации алгоритма поиска в глубину верно соотношение $T_2 < T_1$.*

Из данного утверждения следует, что поиск в глубину в полном орграфе, т.е. $m = n(n-1)/2$, при использовании побитовой матрицы смежностей будет выполнен за время меньшее, чем при его реализации на списках инцидентности для орграфа, у которого $m = n$ с тем же числом вершин. Другими словами, в этом случае время выполнения программной реализации с трудоемкостью $O(n^2)$ меньше времени выполнения программной реализации с трудоемкостью $O(m+n)$ даже если $m = n$.

Утверждение 4. *Если число вершин графа находится в диапазоне от 768 до 4032 ($768 < n < 4032$), то при реализации алгоритма поиска в глубину выполняются следующие соотношения:*

$$T_1 < T_2, \text{ если } p < p^*;$$

$$T_1 > T_2, \text{ если } p > p^*, \text{ где } p = m/n,\ p^* = ([n/64] - 3)/9.$$

Таким образом, при $p = p^*$ объем избыточной информации настолько велик, что векторная обработка необходимой и избыточной информации эквивалентна по времени реализации скалярной обработке только необходимой информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ***Воеводин, В.В.*** Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб. : БХВ, 2002. – 608 с.
2. Принципы построения суперкомпьютеров семейства СКИФ и их реализация / С.В. Абламейко [и др.] // Информатика. – 2004. – № 1. – С. 89–106.
3. ***Иванников, В.П.*** Особенности систем программирования для векторно-конвейерной ЭВМ / В.П. Иванников, С.С. Гайсарян // Кибернетика и вычислительная техника. – Вып. 2. – М. : Наука, 1986. – С. 3–17.
4. ***Евстегнеев, В.Н.*** Графы в программировании: обработка, визуализация и применение / В.Н. Евстегнеев, В.А. Евстегнеев. – СПб. : БХВ, 2003. – 1104 с.

Поступила в редакцию 29.01.2013 г.