№ 4 (325) ТЕХНОЛОГИЯ ТЕКСТИЛЬНОЙ ПРОМЫШЛЕННОСТИ 2010

УДК 677.024: 519.15

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ВЫДЕЛЕНИЯ РАППОРТОВ ТЕКСТИЛЬНЫХ ТЕКСТУР С УЧЕТОМ СЛУЧАЙНЫХ ПОМЕХ

Г.И. БОРЗУНОВ

(Московский государственный текстильный университет им. А.Н. Косыгина)
E-mail: office@msta.ac.ru

В статье описываются результаты, полученные автором при исследовании возможности повышения эффективности параллельного алгоритма выделения повторяющихся фрагментов текстильных текстур.

The results received by the author when researching of the possibility of the efficiency improvement of the parallel algorithm for the pattern distinguish in the repeated fragments of textile textures.

Ключевые слова: параллельный алгоритм, эффективность, раппорт, точечный рисунок.

Пусть точечное изображение представляется целочисленным двумерным массивом A[m][n]. Раппортом будем считать минимальный массив R[mr][nr]: for(i=0; i<mr; i++) for(j=0; j<nr; j++) {ia=(id+i)/%m; ja =(jd+j)/%n; R[i][j] = A[ia][ja];}, где (id, jd) − координаты левого верхнего угла области A[m][n], из которой образуется раппорт R[mr][nr], "%" − здесь и далее представляет собой операцию деления для получения остатка. Как и ранее в [1], ширина nr может меняться от 2 до n/2, а высота mr − от 2 до m/2. Покрытием изображения A[m][n] называется массив V[m][n], который образуется из R[mr][nr] в результате следующих вычислений: for(i=0;i<m;i++) for(j=0;j<n;j++) {if(i>id)ir=(i-id)%mr; else ir=mr-(id-i)%mr; if(j>jd)jr=(j-jd)%nr; else jr=nr-(jd-j)%nr; V[i][j]=R[ir][jr]}. Возможные раппорты оцениваются по их покрытиям исходных изображений с помощью функции несовпадений: /*переменные int mr, nr id, jd, − глобальные */ long int W2(A [m][n]) {long int s=0; int ir, jr; for(int i=0; i<m; i++)for(int j=0; j<n; j++) {if(i>id)ir=(i-id)%mr; else ir=mr-(id-i)%mr; if(j>jd)jr=(j-jd)%nr; else jr=nr-(jd-j)%nr; if(A[i][j]<A[ir][jr]) s+=(A[ir][jr] - A[i][j]); if(A[i][j]>A[ir][jr]) s+=(A[i][j] - A[ir][jr]);} return s;}. Задача выделения раппорта заключается в определении раппорта R[mr][nr], такого что W2(A[m][n],V[m][n]) → min. Второй класс изображений характеризуется тем, что при любом варианте R[mr][nr] значение функции несовпадений больше нуля: W2(A [m][n],V[m][n])>0. Близкие по значениям коды цвета можно рассматривать как случайные отклонения от среднего значения в результате наложения на изображение некоторого шума. Будем, как и прежде [1], рассматривать варианты раппортов R[mr][nr] в порядке увеличения параметра: pr= mr+ nr. Для каждого варианта сочетания значений mr, nr будем рассматривать все возможные соответствия между верхним левым углом раппорта и всеми точками A[m][n], вычисляя для каждого такого соответствия функцию несовпадений. Алгоритм Rpt2 реализует выделение раппорта в изображениях класса 2: int mr, nr, id, jd; /* переменные mr, nr, id, jd − глобальные */ long int Rpt2(A [m][n]) {int pr=0; int prmax=m/2+n/2; long int wmin=255×n×m; long int ww =0; int mrmin=0, nrmin=0, idmin=0, jdmin=0; for(pr=4; pr≤prmax; pr++) {mr=2, nr=pr-mr; if(nr>n/2){nr=n/2; mr=pr-nr;} while((mr≤m/2)&&(nr≥2)) {for(id=0; id<m; id++) for(jd=0; jd<n; jd++) {ww= W2(A [m][n]); if(ww = 0) { return ww;} if(ww < wmin) { mrmin=mr; nrmin=nr; idmin=id; jdmin=jd; wmin=ww; } mr++; nr--; } } if(wmin==255×n×m) {mr=0, nr=0, id=0, jd=0;} else { mr=mrmin; nr=nrmin; id=idmin; jd=jdmin;} return wmin;}. Ранее было показано, что при K=m+n и m=n временная сложность цикла "while((mr≤m/2)&&(nr≥2))" с вложенными циклами функции несовпадений "for(int i=0; i<m; i++)for(int j=0; j<n; j++)" составляет $T3(K)= (K^4–5\times K^3+6\times K^2)/8 \approx O(K^4)$. Если учесть еще два вложенных цикла "for(int id=0; id<m; id++)for(int jd=0; jd<n; jd++"), временная сложность которых равна $T4(K) = (K/2)^2 \approx O(K^2)$, то временная сложность алгоритма Rpt2 оказывается равной $T5=T3\times T4\approx O(K^6)$. Шестая степень полиномиальной оценки временной сложности алгоритма Rpt2 в худшем и в среднем случае является основанием для его реализации с использованием параллельного программирования и многопроцессорных систем. Для этого изменим алгоритм Rpt2, превратив его в алгоритм Rpt2_2, работа которого контролируется двумя параметрами prmin, prmax, определяющими начальное и конечное значение параметра pr, а также массивом fp[p]. Массив fp[p] является характеристическим вектором состояния p процессоров: при fp[ii]==0 – процессор ii остановлен, а при fp[ii]==1 – процессор ii выполняет вычисления согласно алгоритму Rpt2_2. Ниже приводится псевдокод этого алгоритма: int mr[p], nr[p], id[p], jd[p]; /* mr[p], nr[p], id[p], jd[p], − глобальные векторы */ long int Rpt2_2(prmin, prmax, ii, A [m][n]) {int pr; long int wmin=255×n×m; long int ww =0; int mrmin=0, nrmin=0, idmin=0, jdmin=0; for(pr= prmin; pr≤ prmax; pr++) {mr[ii]=2, nr[ii]=pr-mr[ii]; if(nr[ii]>n/2) {nr[ii]=n/2; mr[ii]=pr-nr[ii];} while(mr[ii]≤m/2&&nr[ii]≥2) {ww= W2_2(ii, A[m][n]); if(ww == 0) /*найден раппорт*/ return 0; if(ww < wmin) { mrmin=mr[ii]; nrmin=nr[ii]; idmin=id[ii]; jdmin=jd[ii]; wmin=ww;} mr[ii]++; nr[ii]--;} } if(wmin==255×n×m) {mr[ii]=0, nr[ii]=0, id[ii]=0, jd[ii]=0;} else { mr[ii]=mrmin; nr[ii]=nrmin; id[ii]=idmin; jd[ii]=jdmin;} return wmin;}. В приведенном выше псевдокоде Rpt2_2 используется функция W2_2 которая отличается от функции W2, только тем, что в ней вместо переменных mr, nr, id, jd используются элементы массивов mr[ii], nr[ii], id[ii], jd[ii]. Как и прежде, если после завершения Rpt2_2 значения mr[ii], nr[ii] не равны нулю, то R[mr[ii]][nr[ii]] = A[mr[ii]][nr[ii]]. При равенстве нулю значений mr[ii], nr[ii] для данного A [m][n] раппорт не определен в интервале prmin ≤mr[ii]+nr[ii]≤ prmax. При использовании p= m+n-3 процессоров простейший подход к распараллеливанию описанного выше поиска раппорта на основе применения алгоритма Rpt2_2 реализуется по схеме, предложенной в [1]: каждому из доступных процессоров передается одно из значений параметра pr, для которого осуществляется указанный поиск. Такое количество процессоров для обработки реальных изображений с использованием систем с разделенной памятью (кластеров), как правило, оказывается недоступным. Ниже впервые описывается разработанный автором параллельный алгоритм Rpt6, реализующий указанную схему для случая p<m+n-3. Псевдокод алгоритма Rpt6 состоит из следующих четырех шагов. 1. Работа центрального процессора начинается со следующих действий: вводятся параметры точечного рисунка m,n и сам точечный рисунок A[m][n]; for(i=0;i<p; i++){ fp[i]=0;} pmr=m; pnr=n; pii= p+1; pwmin=255×n×m. 2. Центральный процессор определяет и передает дос-

тупным процессорам диапазоны значений параметра pr (prmin[i], prmax[i]) и активизирует их работу: {int dpr=((m+n-3)/p-1); /*целочисленное деление выполняется с отбрасыванием дробной части*/ mpr==((m+n-3)%p); / prmin[0]=4; if(mpr>0) {prmax[0]=prmin[0]+ dpr +1; mpr--;} else { prmax[0]=prmin[0]+ dpr;} for(ii=1;ii <(m+n-3); ii++){ prmin[ii]=prmax[ii-1]+1; if(mpr>0) {prmax[ii]=prmin[ii]+dpr+1; mpr--;} else { prmax[ii]=prmin[ii]+dpr;}передать в сообщении значения prmin[ii], prmax[ii], ii и A[m][n] процессору PR[ii] и активизировать его работу; fp[ii]=1;}. 3. Активизированные процессоры осуществляют проверку существования раппортов, параметр pr которых заключен в заданном диапазоне: **Parallel Start** {for(ii=1; ii<p; ii++) {PR[ii]: читать из сообщения параметры prmin[], prmax[], ii, A[m][n]; PR[ii]: ww= =Rpt2_2(prmin, prmax, ii, A[m][n]); PR[ii]: передать в сообщении центральному процессору параметры ii, ww; стоп;} **Parallel End.** 4. Центральный процессор ожидает сообщение и, получив сообщение от ii-го процессора, выполняет действия: while(fp[n]! =(0, 0, …, 0)) {читать из сообщения параметры ii, ww; if(pwmin>ww){ pwmin=ww; pii=ii; pmr=mr[ii]; pnr=nr[ii];} else { if((pwmin==ww) && (pmr+pnr> >mr[ii]+nr[ii])) {pii=ii; pmr=mr[ii]; pnr=nr[ii];} fp[i]=0;}} if(pmr ≤m/2&&pnr≤n/2) {прямоугольный фрагмент массива A[m][n], левый верхний угол которого определяется координатами (id[pii], jd[pii]), а правый нижний – координатами (id[pii]+(pmr-1), jd[pii]+(pnr-1)), представляет собой найденный раппорт минимальных размеров; функция несовпадений для этого раппорта равна значению pwmin; } else{раппорт не определен}, стоп. Описанный выше алгоритм решает задачу параллельного поиска раппортов точечных изображений класса 2 при p<m+n-3, но не обеспечивает равномерного распределении вычислительной нагрузки по доступным процессорам. Равномерная вычислительная нагрузка обеспечивается алгоритмом Rpt6_2, который использует prmin[], prmax[], рассчитанные по алгоритму Balance6_2(m, n,/*m,n – размеры изображения*/ p, /*число доступных процессоров*/ prmin[p], prmax[p]) {int q=(m+n)/2; int dpn=((q-3)*(q-2))/2/p; prmin[0]=4; prmax[0] = ОКРУГЛИТЬ(КОРЕНЬ(2*dpn - 0,25)+2,5); for(i=1; i<=p; i++){ prmin[i]= prmax[i-1]+1; prmax[i]= ОКРУГЛИТЬ (КОРЕНЬ(2*(49750,3-0,25+ (prmax[i-1] - 3)*(prmax[i-1]-2)/2))+2,5);}. В табл. 1 приводятся результаты вычислительного эксперимента, выполненного при p=10, m=1200, n=800.

Таблица 1

Алгоритм	Коэффициент ускорения		Коэффициент эффективности	
	в худшем случае	в среднем случае	в худшем случае	в среднем случае
Rpt6	5,300931253	7,55E+00	0,530093125	7,55E-01
Rpt6_2	9,930199601	1,00E+01	0,99301996	1,00E+00

ВЫВОДЫ

1. Ускорение и эффективность алгоритма Rpt6_2 превышают в худшем случае на 87,3% и в среднем случае на 32,5% аналогичные показатели алгоритма Rpt6.

2. Равномерность распределения вычислительной нагрузки существенно влияет на эффективность параллельных алгоритмов.

ЛИТЕРАТУРА

1. *Борзунов Г. И.* Исследование распараллеливания алгоритма выделения раппортов точечных изображений // Вторая регион. научн.-техн. конф.: Применение многопроцессорных суперкомпьютеров в исследованиях, наукоемких технологиях и учебной работе. – Иваново: Ивановская государственная текстильная академия, 2008, 22-23 мая.

Рекомендована кафедрой информационных технологий и компьютерного дизайна. Поступила 20.10.09.