УДК 681.3

ПРЫХОЖЫ А. А., КАРАСІК А. М.

КААПЕРАТЫЎНЫЯ БЛОЧНА-ПАРАЛЕЛЬНЫЯ АЛГАРЫТМЫ РАШЭННЯ ЗАДАЧ НА ШМАТ'ЯДРАВЫХ СІСТЭМАХ

Беларускі нацыянальны тэхнічны універсітэт

Разглядаецца праблема балансавання вылічальнай нагрузкі на ядрах шмат'ядравай сістэмы і павышэння эфектыўнасці ўзаемадзеяння патокаў у шматпаточных праграмных прыкладанняў. Прапанаваны кааператыўныя блочна-паралельныя алгарытмы рашэння складаных задач, якія паддаюцца падзелу на часткі. Яны памяншаюць колькасць перадач кіравання паміж патокамі, скарачаюць даўжыню крытычнага шляху ў паралельнай рэалізацыі, павышаюць загружанасць ядраў.

Увядзенне

Праблема распрацоўкі эфектыўных шматпаточных праграмных прыкладанняў для шмат'ядравых сістэм з'яўляецца навукова актуальнай і практычна запатрабаванай у навукова-тэхнічных галінах. Паралельнае шматпаточнае праграмаванне выконваецца для вырашэння многіх задач, якія маштабуюцца, напрыклад, задачы рашэння сістэмы лінейных алгебраічных раўнанняў (СЛАР) [1]. Праграмаванне і выкарыстанне шмат'ядравых сістэм падтрымліваецца аперацыйнымі сістэмамі з выцясняючай і кааператыўнай шматзадачнасцю. У працы [2] даследаваныя метады рэалізацыі шматпаточных прыкладанняў на шмат'ядравых сістэмах, якія кіруюцца аперацыйнай сістэмай з выцясняючай шматзадачнасцью. У працы [3] паказана, што выкарыстанне кааператыўнай шматзадачнасці з'яўляецца пераважнай у параўнанні з выцясняючай шматзадачнасцю і што кааператыўная мадэль выканання патокаў дазваляе атрымаць значнае паскарэнне на шмат'ядравай сістэме. У дадзеным артыкуле прапануюцца кааператыўныя блочна-паралельныя алгарытмы рашэння задач, якія паддаюцца падзелу на часткі, і праводзіцца эксперыментальнае супастаўленне прапанаваных і вядомых алгарытмаў па шырокаму спектру параметраў і залежнасцяў.

Блочна-паралельныя алгарытмы Гаўса рашэння СЛАР

Метад Гаўса вырашае СЛАР выгляду $\boldsymbol{A} \times \boldsymbol{x} = \boldsymbol{b}$, дзе $\boldsymbol{A}$ – лічбавая матрыца памерам $M \times M$; $\boldsymbol{x}$ – вектар зменных памерам M; $\boldsymbol{b}$ – лічбавы вектар памерам M. Прамы ход прыводзіць матрыцу $\boldsymbol{A}$ да трохкутнаму выгляду. Зваротны ход знаходзіць значэння зменных $\boldsymbol{x}$. Блочна-паралельныя алгарытмы Гаўса [1] арыентаваны на паралельную рэалізацыю на шматпрацэсарных сістэмах (мал. 1). Яны разбіваюць матрыцу $\boldsymbol{A}$ на блокі $\boldsymbol{A}^t$, $t = 0, \ldots, T-1$ памерам $N \times M$ кожны, дзе $N = M / T$, і разбіваюць вектары $\boldsymbol{x}$ і $\boldsymbol{b}$ на блокі $\boldsymbol{x}^t$ і $\boldsymbol{b}^t$ адпаведна памерам N. Алгарытм μ1 ўключае ў паток t, які прымае значэнні ад 0 да $T-1$, радкі $r = 0 + t \times N$, $1 + t \times N \ldots N - 1 + t \times N$ такім чынам, што ў паток 0 ўваходзяць радкі 0, ..., $N-1$, ў паток 1 ўваходзяць радкі N, ..., $2N-1$ і г. д. У якасці актыўнага радка алгарытм μ1 спачатку паслядоўна выбірае радкі патоку 0, затым радкі патоку 1 і г. д. да патоку $T-1$ (мал. 1 злева). Алгарытм μ2 ўключае ў паток t радкі $r = t, T + t, \ldots, (N-1) \times T + t$ такім чынам, што ў паток 0 ўваходзяць радкі 0, T, ..., $(N-1) \times T$, у паток 1 ўваходзяць радкі 1, $T+1$, ..., $(N-1) \times T + 1$ і г. д. У якасці актыўнага радка алгарытм μ2 спачатку паслядоўна выбірае нулявыя радкі патокаў 0, ..., $T-1$, затым першыя радкі патокаў 0, ..., $T-1$, і г. д. да радкоў з нумарамі $N-1$ (мал. 1 справа).

На мал. 1 аперацыя К(t, k) пералічвае актыўны радок $v = t \times N + k$ матрыцы $\boldsymbol{A}$ ў патоку t па першым ненулявым элеменце з нумарам v ў радку v: $a_{vj} = a_{vj}/a_{vv}$ і $b_v = b_v/a_{vv}$ для $j = v, \ldots, M-1$. Аперацыя П(MyT, t, k, i) пералічвае пасіўны радок $v = MyT \times N + i$ матрыцы $\boldsymbol{A}$ ў пато-

```
for(p = 0; t < T; t++) {
  for(k = 0; k < N; k++) {
    if(myT==t) {
      K(t,k);
      for(d=t+1; d<T; d++) SEND(t,k,d);
      for(i=k+1; i<N; i++) П(t,t,k,i);
    } else if(myT>t) {
      RECEIVE(myT,k,t);
      for(i=0; i<N; i++) П(myT,t,k,i);
}}}
```

μ1

```
for(k = 0; k < N; k++) {
  for(p = 0; t < T; t++) {
    if(myT == t) {
      K(t,k);   SEND(t,k,all);
      for(i=k+1; i<N; i++) П(t,t,k,i);
    } else if(myT<t) {
      RECEIVE(myT,k,t);
      for(i=k+1; i<N; i++) П(myT,t,k,i);
    } else if(myT>t) {
      RECEIVE(myT,k,t);
      for(i=k; i<N; i++) П(myT,t,k,i);
}}}
```

μ2

Малюнак 1. - Блочна-паралельныя алгарытмы μ1 і μ2 для аднаго патоку *MyT* (прамы ход)

№	ЛП 0			ЛП 1		
1	Т0	К0	С0			
2	Т0	П0	С1	Т1	П0	С2
3	Т2	П0	С4	Т1	П0	С3
4	Т2	П0	С5	Т3	П0	С6
5	Т0	К1	С1	Т3	П0	С7
6	Т2	П1	С4	Т1	П1	С2
7	Т2	П1	С5	Т1	П1	С3
8				Т3	П1	С6
9				Т3	П1	С7
10				Т1	К2	С2
11	Т2	П2	С4	Т1	П2	С3
12	Т2	П2	С5	Т3	П2	С6
13				Т3	П2	С7
14				Т1	К3	С3
15	Т2	П3	С4	Т3	П3	С6
16	Т2	П3	С5	Т3	П3	С7
17	Т2	К4	С4			
18	Т2	П4	С5	Т3	П4	С6
19	Т2	К5	С5	Т3	П4	С7
20				Т3	П5	С6
21				Т3	П5	С7
22				Т3	К6	С6
23				Т3	П6	С7
24				Т3	К7	С7

μ1

№	ЛП 0			ЛП 1		
1	Т0	К0	С0			
2	Т0	П0	С4	Т1	П0	С1
3	Т2	П0	С2	Т1	П0	С5
4	Т2	П0	С6	Т3	П0	С3
5				Т3	П0	С7
6				Т1	К1	С1
7	Т0	П1	С4	Т1	П1	С5
8	Т2	П1	С2	Т3	П1	С3
9	Т2	П1	С6	Т3	П1	С7
10	Т2	К2	С2			
11	Т2	П2	С6	Т1	П2	С5
12	Т0	П2	С4	Т3	П2	С3
13				Т3	П2	С7
14				Т3	К3	С3
15	Т0	П3	С4	Т3	П3	С7
16	Т2	П3	С6	Т1	П3	С5
17	Т0	К4	С4			
18	Т2	П4	С6	Т1	П4	С5
19				Т3	П4	С7
20				Т1	К5	С5
21	Т2	П5	С6	Т3	П5	С7
22	Т2	К6	С6			
23				Т3	П6	С7
24				Т3	К7	С7

μ2

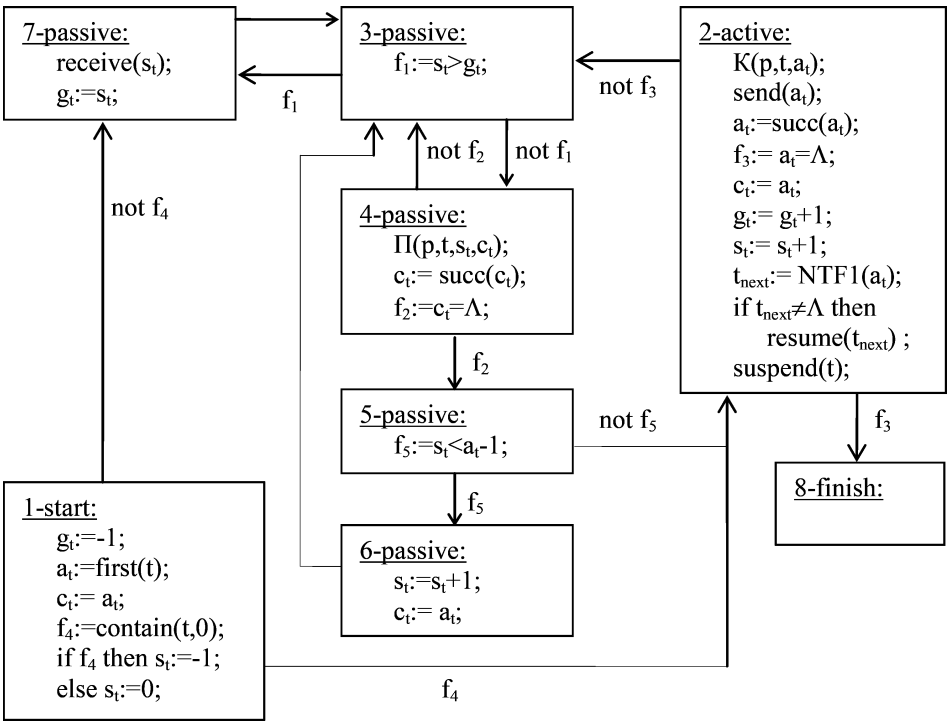
Малюнак 2. - Крокі працы алгарытмаў μ1 і μ2 на 8 радках, 4 патоках, 2 працэсарах (прамы ход)

ку t па актыўнаму радку $u = t \times N + k$: $a_{vj} = a_{vj} - a_{vu} \times a_{uj}$ і $b_v = b_v - a_{vu} \times b_u$ для $j = u, ..., M - 1$. Працаёмкасць аперацый К(t, k) і П(MyT, t, k, i) лінейна меншае з павелічэннем нумары актыўнага патоку t і нумары актыўнага радка k, які належыць патоку t. Як паказана ў працы [2], вылічальная нагрузка нераўнамерная па патоках, прычым алгарытм μ1 дае больш нераўнамернае размеркаванне ў параўнанні з алгарытмам μ2.

Пры выкананні, патокі прызначаюцца на P працэсараў, прычым адзін працэсар рэалізуе T/P патокаў. Для алгарытмаў μ1 і μ2 мы выкарыстоўваем адно і тое ж прызначэнне: на працэсар $p \in \{0, ..., P - 1\}$ прызначаюцца патокі з нумарамі $p, p + P, p + 2P, ..., (K - 1) \times P + p$, дзе $K = T/P$. Такое прызначэнне згладжвае нераўнамернае размеркаванне вылічальнай нагрузкі па патоках. Мал. 2 ілюструе крокі працы алгарытмаў μ1 і μ2 пры вырашэнні СЛАР з 8 радкоў, прызначаных на 4 патокі, якія выконваюцца на 2 працэсарах. Тут T_i – бягучы паток; K_j і Пj – аперацыі, якія выконваюцца і выкарыстоўваюць радок j; C_k – радок, над якшм выконваюцца аперацыі.

Кааператыўны блочна-паралельны алгарытм μ1к

Асноўным прынцыпам пабудовы кааператыўнага алгарытму [1] з'яўляецца мінімізацыя колькасці перадач кіравання паміж патокамі на адным працэсары. Пры прызначэнні на адзін працэсар K патокаў, мінімальная колькасць перадач кіравання роўная $K - 1$. Гэты прынцып



Малюнак 3. - Кааператыўны блочна-паралельны алгарытм μ1к рашэння СЛАР (прамы ход)

добра спалучаецца з размеркаваннем радкоў па патоках у алгарытме μ1, дзякуючы якому паток t пасылае паведамленні патокам $h > t$ і ня пасылае паведамленні патокам $h < t$. Кааператыўны алгарытм будуецца такім чынам, што кожны з патокаў цалкам выконвае сваю працу, а затым перадае кіраванне наступнаму патоку з большым нумарам, прызначанаму на той жа працэсар. Неабходнай умовай выканання работы патокам з'яўляецца назапашванне ўсіх радкоў ад іншых патокаў, якія папярэднічаюць радкам дадзенага патоку. Патокі на адным працэсары выконваюцца паслядоўна, на розных працэсарах – паралельна.

Дыяграма высокаўзроўневага аўтамата, якая апісвае паводзіны аднаго актыўнага патоку t, што функцыянуе згодна з алгарытмам μ1к, выконваецца на працэсары p і ўзаемадзейнічае з іншымі патокамі, паказаная на мал. 3. Дыяграма ўключае 8 станаў, якія функцыянуюць у 4 рэжымах: *start*, *active*, *passive* і *finish*. Стан 1-*start* ініцыялізуе паток t. Стан 2-*active* выконвае аперацыі над актыўным радком a_t. Станы 3-*passive* ... 7-*passive* выконваюць аперацыі над пасіўным радком c_t, каэфіцыенты якога пералічваюцца па радку s_t. Стан 8-*finish* з'яўляецца канчатковым ў патоку t.

У пачатковым стане 1-*start* зменнай g_t, якая прадстаўляе на бягучы момант часу глабальна актыўны радок, прысвойваецца пустое значэнне –1. Далей, у наступных выпадках, радок g_t генеруецца патокам t на працэсары p з дапамогай аперацыі K(p, t, k) або прымаецца ад іншага патоку, які выконваецца на працэсары p або на іншых працэсарах. Зменнай a_t, ($a_t > g_t$), якая вызначае радок, што павінен стаць наступным актыўным радком ў патоку t, прысвойваецца найменшы ў патоку t радок *first*(t). Заўважым, што радку a_t можа спатрэбіцца пэўны час каб стаць актыўным, паколькі яму можа спатрэбіцца пералік ад іншых папярэдніх радкоў з іншых патокаў, якія стануць актыўнымі да таго як актыўным стане радок a_t. Першым пасіўным радком c_t патоку t з'яўляецца пачатковы радок *first*(t). Калі паток t ўтрымлівае радок з нулявым нумарам (у гэтым выпадку выконваецца прэдыкат *contain*(t, 0)), зменнай $f4$ прысвойваецца значэнне *true*, а зменнай s_t прысвойваецца пустое значэнне –1, пасля чаго выконваецца пераход аўтамата ў стан 2-*active*. У адваротным выпадку зменнай $f4$ прысвойваецца значэнне *false*, зменнай s_t прысвойваецца значэнне 0, а аўтамат пераходзіць у стан 7-*passive*.

У стане 2-*active* радок a_t нармалізуецца аперацыяй K(p, t, a_t) і пасылаецца іншым патокам з мэтай пераліку прызначаных на іх радкоў. У якасці новага a_t выбіраецца наступны

```
function NextThreadF(Th, Pr) {
    for each p∈P { πp:=-1; }
    for i:=1,…,M {
        t := Thi; p := Prt;
        if πp=-1 then { πp:=i; } else { NTF(πp):=t; πp:=i; }
    }
    for each p∈P { NTF(πp):=-1; }
    return NTF;
}
```

Малюнак 4. - Функцыя вылічэння вектара *NTF* патокаў, якім перадаецца кіраванне пасля выканання аперацыі K(p, t, a_t)

радок, уключаны ў t, які становіцца таксама пасіўна пералічваемым радком c_t. Нумары глабальна актыўнага радка g_t і радка s_t павялічваюцца на 1, а наступным актыўным патокам t_{next} на працэсары p становіцца паток $NTF1(a_t)$. Аператар $resume(t_{next})$ аднаўляе паток t_{next} калі гэта непусты паток, а аператар $suspend(t)$ прыпыняе паток t. Калі радок a_t з'яўляецца пустым Λ, то зменнай $f3$ прысвойваецца значэнне *true*, а аўтамат пераходзіць у канчатковы стан 8-*finish*, у адваротным выпадку аўтамат пераходзіць у стан 3-*passive*. Функцыя *NextThreadF* вылічэння нумароў патокаў *NTF* прадстаўлена на мал. 4. Для СЛАР з 8 радкоў, прызначаных на 4 патокі, што выконваюцца на 2 працэсарах, вектар $NTF1 = \{0, 2, 1, 3, 2, –1, 3, –1\}$ вылічаецца выклікам функцыі з параметрамі $Th = \{0, 0, 1, 1, 2, 2, 3, 3\}$ і $Pr = \{0, 1, 0, 1\}$. Элемент Th_i вектара *Th* ёсць нумар патоку, які змяшчае радок i, а элемент Pr_t вектара *Pr* ёсць нумар працэсара, на якім выконваецца паток t. Першы элемент 0 на вектары $NTF1$ азначае, што пасля нармалізацыі радка 0 на працэсары 0 кіраванне застаецца ў патоку 0, а другі элемент 2 азначае, што пасля нармалізацыі радка 1 кіраванне перадаецца патоку 2 на тым жа працэсары 0. Значэнне –1 азначае завяршэнне вылічэнняў на адпаведным працэсары.

У стане 3-*passive* булевай зменнай $f1$ прысвойваецца значэнне *true*, калі нумар радка s_t больш нумара глабальна актыўнага радка g_t, і аўтамат пераходзіць у стан 7-*passive* з мэтай чакання прыёму радка $s_t(g_t)$ ад іншых патокаў. Калі выконваецца роўнасць $s_t = g_t$, то радок s_t ўжо з'яўляецца глабальна актыўным, зменнай $f1$ прысвойваецца значэнне *false*, і аўтамат пераходзіць у стан 4-*passive*. У стане 4-*passive* аперацыя $П(p, t, s_t, c_t)$ пералічвае пасіўны радок c_t па радку s_t, а наступны пералічваемы радок вызначаецца аперацыяй $succ(c_t)$. Калі гэта пусты радок Λ, булева зменная $f2$ атрымлівае значэнне *true*, і аўтамат пераходзіць у стан 5-*passive*. У адваротным выпадку аўтамат вяртаецца ў стан 3-*passive*. У стане 5-*passive*, калі радок s_t з'яўляецца апошнім, па якому пералічваецца радок a_t, булева зменная $f5$ атрымлівае значэнне *false* і аўтамат пераходзіць у стан 2-*active*. У адваротным выпадку аўтамат пераходзіць у стан 6-*passive*, дзе нумар радка s_t павялічваецца на адзінку і прысвойваецца нумары першага пералічваемага радка c_t, пры гэтым аўтамат вяртаецца ў стан 3-*passive*. У стане 7-*passive* паток t чакае з дапамогай аператара $receive(s_t)$ прыёму актыўнага радка s_t ад іншага патоку. Пры прызначэнні патокаў на адзін працэсар, аперацыі пасылкі-прыёму паведамленняў выконваюцца хутчэй, калі на розныя працэсары – больш павольна.

Асаблівасцю працы алгарытму μ1к у пасіўных станах 3-*passive* – 7-*passive* з'яўляецца паслядоўны перабор актыўных радкоў s_t, пачынаючы з 0 і канчаючы $a_t – 1$ з мэтай пераліку пасіўных радкоў ад a_t да пустога радка Λ.

Крокі працы алгарытму μ1к на СЛАР памерам 8 × 8, вырашаемай 4 патокамі, выканваемымі на 2 працэсарах, паказаны на мал. 5 злева. Колькасць крокаў скарацілася на 1 у параўнанні з алгарытмам μ1, пры гэтым, у адрозненне ад алгарытму μ1, які выконвае 2 перадачы кіравання паміж патокамі на працэсары ЛП0 і 7 перадач кіравання на працэсары ЛП1, алгарытм μ1к выконвае толькі па адной перадачы кіравання на кожным з працэсараў. Высокаўзроўневы аўтамат, які апісвае паводзіны патоку пры зваротным ходзе па матрыцы $\boldsymbol{A}$, будуецца аналагічным чынам.

Кааператыўны блочна-паралельны алгарытм μ2к

Асноўным прынцыпам пабудовы кааператыўнага алгарытму μ2к з'яўляецца мінімізацыя колькасці пераходаў ад пераліку каэфіцы-

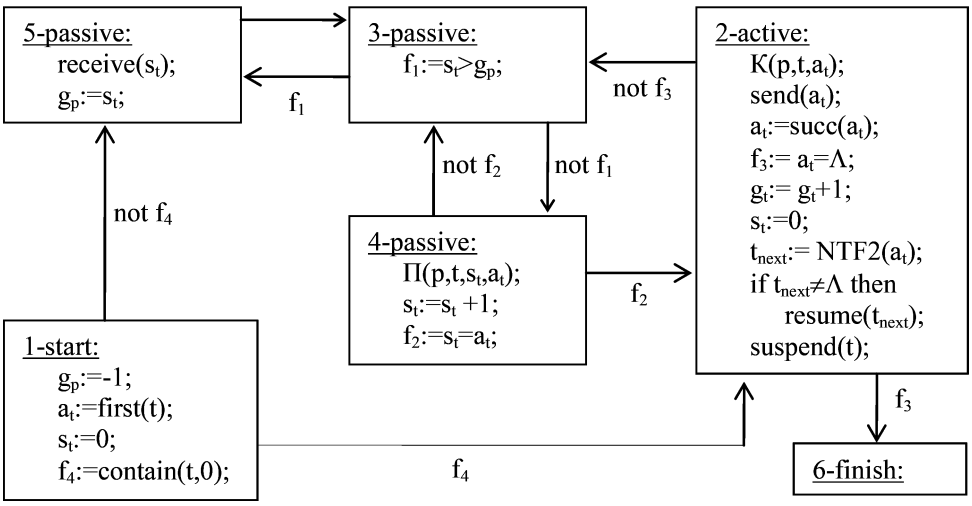
№	ЛП 0			ЛП 1		
1	Т0	К0	С0			
2	Т0	П0	С1	Т1	П0	С2
3	Т0	К1	С1	Т1	П0	С3
4	Т2	П0	С4	Т1	П1	С2
5	Т2	П0	С5	Т1	П1	С3
6	Т2	П1	С4	Т1	К2	С2
7	Т2	П1	С5	Т1	П2	С3
8	Т2	П2	С4	Т1	К3	С3
9	Т2	П2	С5	Т3	П0	С6
10	Т2	П3	С4	Т3	П0	С7
11	Т2	П3	С5	Т3	П1	С6
12	Т2	К4	С4	Т3	П1	С7
13	Т2	П4	С5	Т3	П2	С6
14	Т2	К5	С5	Т3	П2	С7
15				Т3	П3	С6
16				Т3	П3	С7
17				Т3	П4	С6
18				Т3	П4	С7
19				Т3	П5	С6
20				Т3	П5	С7
21				Т3	К6	С6
22				Т3	П6	С7
23				Т3	К7	С7

μ1к

№	ЛП 0			ЛП 1		
1	Т0	К0	С0			
2	Т2	П0	С2	Т1	П0	С1
3				Т1	К1	С1
4	Т2	П1	С2	Т3	П0	С3
5	Т2	К2	С2	Т3	П1	С3
6	Т0	П0	С4	Т3	П2	С3
7	Т0	П1	С4	Т3	К3	С3
8	Т0	П2	С4	Т1	П0	С5
9	Т0	П3	С4	Т1	П1	С5
10	Т0	К4	С4	Т1	П2	С5
11	Т2	П0	С6	Т1	П3	С5
12	Т2	П1	С6	Т1	П4	С5
13	Т2	П2	С6	Т1	К5	С5
14	Т2	П3	С6	Т3	П0	С7
15	Т2	П4	С6	Т3	П1	С7
16	Т2	П5	С6	Т3	П2	С7
17	Т2	К6	С6	Т3	П3	С7
18				Т3	П4	С7
19				Т3	П5	С7
20				Т3	П6	С7
21				Т3	К7	С7

μ2к

Малюнак 5. - Крокі працы алгарытмаў μ1к і μ2к на 8 радках, 4 патоках, 2 працэсарах (прамы ход)



Малюнак 6. - Кааператыўны блочна-паралельны алгарытм μ2к рашэння СЛАР (прамы ход)

ентаў аднаго радка да пераліку каэфіцыентаў іншага радка. Пры прызначэнні на адзін працэсар K патокаў, якія змяшчаюць сумесна $S = M/P$ радкоў, міні-мальная колькасць пераходаў роўная $S - 1$. Гэтыя пераходы паміж радкамі могуць павялічыць колькасць перадач кіравання паміж патокамі у параўнанні з алгарытмам μ1к. Ўказаны прынцып добра спалучаецца з размеркаваннем радкоў па патоках у алгарытме μ2, дзякуючы якому вылічэнні эфектыў на размяркоўваюцца і распаралельваюцца на працэсарах. Алгарытм μ2к будуецца такім чынам, што цалкам апрацоўваецца бягучы радок матрыцы СЛАР, а затым выконваецца пераход да апрацоўкі наступнага радка на дадзеным працэсары, з магчымай перадачай кіравання іншаму патоку. Неабходнай умовай завяршэння пераліку бягучага радка з'яўляецца назапашванне ўсіх папярэдніх радкоў, пералік якіх завершаны.

Дыяграма высокаўзроўневага аўтамата, якая апісвае паводзіны алгарытму μ2к ў патоку t на працэсары p, паказаная на мал. 6. Дыяграма ўключае 6 станаў, якія функцыянуюць у 4 рэжымах: *start*, *active*, *passive* і *finish*. У пачатковым стане 1-*start* глабальна актыўным радком g_t становіцца пусты радок (з нумарам –1),

актыўным радком a_t становіцца радок *first*(t), радком s_t, які выкарыстоўваецца для пераліку іншых наступных радкоў, становіцца радок з нумарам 0. Пераход аўтамата ў наступны стан вызначаецца прэдыкатам *contain*(t, 0) і зменнай $f4$, прымаючай значэнне *true*, калі радок з нумарам 0 ўключаны ў паток t.

У стане 2-*active* радок a_t нармалізуецца аперацыяй К(p, t, a_t) і пасылаецца іншым патокам з мэтай пераліку радкоў з большымі нумарамі. Новым значэннем a_t становіцца наступны радок, уключаны ў паток t, а нумар глабальна актыўнага радка g_t павялічваецца на 1. Значэннем s_t становіцца радок з нумарам 0. Наступны актыўны паток t_{next} на працэсары p выбіраецца аператарам $NTF2(a_t)$ у залежнасці ад нумара радка a_t. Паток t_{next} аднаўляецца аператарам *resume*(t_{next}), калі гэта не пусты паток, а паток t прыпыняецца аператарам *suspend*(t). Пераход аўтамата ў наступны стан 3-*passive* або 8-*finish* вызначаецца булевай зменнай $f3$, якая прымае значэнне *true*, калі радок a_t з'яўляецца пустым (Λ). Для разгляданага прыкладу, вектар $NTF2 = \{2, 3, 0, 1, 2, 3, -1, -1\}$ нумароў патокаў, якім перадаецца кіраванне пасля завяршэння апрацоўкі радкоў, вызначаецца ў выніку выкліку функцыі *NextThreadF* з параметрамі $Th = \{0, 1, 2, 3, 0, 1, 2, 3\}$ і $Pr = \{0, 1, 0, 1\}$.

У стане 3-*passive*, калі $s_t > g_t$, аўтамат пераходзіць у стан 5-*passive*, калі ж $s_t = g_t$, аўтамат пераходзіць у стан 4-*passive*. У стане 4-*passive* аперацыя П(p, t, s_t, a_t) пералічвае радок a_t па радку s_t. Нумар наступнага радка s_t, павялічваецца на 1. Калі новы радок s_t супадае с радком a_t, пералік радка a_t ад папярэдніх радкоў скончаны і аўтамат пераходзіць у стан 2-*active*. У адваротным выпадку пералік не скончаны, аўтамат вяртаецца ў стан 3-*passive*. У стане 5-*passive*, аўтамат чакае радок s_t, пералічаны іншымі патокамі. У выпадку атрымання такога радка аператарам *receive*(s_t), аўтамат вяртаецца ў стан 3-*passive*.

Крокі працы алгарытму μ2к на СЛАР памерам 8 × 8, які выконваецца 4 патокамі на 2 працэсарах, паказаны на мал. 5 справа. Колькасць крокаў скарацілася на 2 у параўнанні з алгарытмам μ1к, аднак, у адрозненне ад μ1к, які выконвае па 1-й перадачы кіравання паміж патокамі на кожным працэсары, алгарытм μ2к выконвае больш, а менавіта, па 3 перадачы на кожным працэсары. У той жа час, колькасць перадач значна скарацілася ў параўнанні з алгарытмам μ2, які выконвае 7 перадач на ЛП0 і 9 перадач на ЛП1. Высокаўзроўневы аўтамат, які апісвае паводзіны патоку пры зваротным ходзе па матрыцы СЛАР, будуецца аналагічным чынам.

Эксперыментальнае даследаванне кааператыўных блочна-паралельных алгарытмаў

Найважнейшымі параметрамі шматпаточных прыкладанняў з'яўляюцца: колькасць выконваемых макра аперацый К(p, t, a_t) і П(p, t, s_t, a_t); даўжыня крытычнага шляху ў ліку макра аперацый (СР); каэфіцыент распаралельвання, ацэньваны як агульная колькасць макра аперацый падзеленае на даўжыню крытычнага шляху (Prl); сярэдняя загрузка працэсара (Ld%); сярэдняя колькасць паведамленняў паміж парай патокаў (MbT); сярэдняя колькасць паведамленняў паміж парай працэсараў (MbP); сярэдняя колькасць паведамленняў ўнутры аднаго працэсара (MonP); сярэдняя колькасць перадач кіравання паміж патокамі на адным працэсары (СТР).

Праведзены тры серыі эксперыментаў, якія выяўляюць змену параметраў алгарытмаў μ1, μ2, μ1к і μ2к ў залежнасці ад: памеру матрыцы СЛАР, колькасці патокаў і колькасці працэсараў. Вынікі прадстаўлены ў табл. 1–6. Павелічэнне радкоў СЛАР з 8 да 128 павялічвае лік макра аперацый з 72 да 16512 (табл. 1). Пры выкарыстанні 8 патокаў і 4 працэсараў, даўжыня крытычнага шляху павялічваецца з 36 да 5676 для алгарытму μ1 і з 30 да 5670 для алгарытму μ1к. Для алгарытмаў μ2 і μ2к павелічэнне склала з 36 да 4416, і з 30 да 4230 адпаведна. Як следства, каэфіцыент распаралельвання і загрузка працэсараў вышэй у μ1к у параўнанні з μ1 на 0.1% – 20%, і вышэй у μ2к у параўнанні з μ2 на 4.4% – 20%. Па гэтых параметрах μ2к пераўзыходзіць μ1к да 34%, прычым перавага ўзрастае з ростам памеру СЛАР. Лік паведамленняў паміж парай патокаў і ўнутры аднаго працэсара ўзрастае з ростам памеру СЛАР для ўсіх алгарытмаў (табл.2). Яно каля 2 разоў вышэй для μ2 і μ2к у параўнанні з μ1 і μ1к. Лік паведамленняў паміж парай працэсараў расце хутчэй для μ2 і μ2к, чым для μ1 і μ1к. Лік перадач кіравання

Т а б л і ц а 1. **Залежнасць параметраў CP, Prl і Ld ад колькасці радкоў у матрыцы СЛАР для алгарытмаў μ1, μ2, μ1к і μ2к, 8 патокаў і 4 працэсараў**

Лік радкоў СЛАР	Лік макра аперацый	Алгарытмы / Параметры											
		μ1			μ2			μ1к			μ2к		
		CP	Prl	Ld%	CP	Prl	Ld%	CP	Prl	Ld%	CP	Prl	Ld%
8	72	36	2.00	50.0	36	2.00	50.0	30	2.40	60.0	30	2.40	60.0
16	272	104	2.62	65.4	104	2.62	65.4	98	2.78	69.4	86	3.16	79.1
32	1056	372	2.84	71.0	336	3.14	78.6	366	2.89	72.1	294	3.59	89.8
64	4160	1436	2.90	72.4	1184	3.51	87.8	1430	2.91	72.7	1094	3.80	95.1
128	16512	5676	2.91	72.7	4416	3.74	93.5	5670	2.91	72.8	4230	3.90	97.6

Т а б л і ц а 2. **Залежнасць параметраў MbT, MbP, MonP і CTP ад колькасці радкоў у матрыцы СЛАР для алгарытмаў μ1, μ2, μ1к і μ2к, 8 патокаў і 4 працэсараў**

Лік радкоў СЛАР	Алгарытмы / Параметры									
	μ1 і μ1к			μ1	μ1к	μ2 і μ2к			μ2	μ2к
	MbT	MbP	MonP	CTP	CTP	MbT	MbP	MonP	CTP	CTP
8	1	3	2	8	2	1	3	2	8	2
16	2	6	4	18	2	3	6	6	50	6
32	4	12	8	38	2	7	15	14	230	14
64	8	24	16	78	2	15	31	30	974	30
128	16	48	32	158	2	31	63	62	3998	62

Т а б л і ц а 3. **Залежнасць параметраў CP, Prl і Ld ад колькасці патокаў для алгарытмаў μ1, μ2, μ1к і μ2к, 64 радкоў у матрыцы СЛАР (4160 макра аперацый) і 4 працэсараў**

Лік патокаў	Алгарытмы / Параметры											
	μ1			μ2			μ1к			μ2к		
	CP	Prl	Ld%	CP	Prl	Ld%	CP	Prl	Ld%	CP	Prl	Ld%
4	1814	2.29	57.3	1184	3.51	87.8	1814	2.29	57.3	1094	3.80	95.1
8	1436	2.90	72.4	1184	3.51	87.8	1430	2.91	72.7	1094	3.80	95.1
16	1256	3.31	82.8	1184	3.51	87.8	1238	3.36	84.0	1094	3.80	95.1
32	1184	3.51	87.8	1184	3.51	87.8	1142	3.64	91.1	1094	3.80	95.1
64	1184	3.51	87.8	1184	3.51	87.8	1094	3.80	95.1	1094	3.80	95.1

Т а б л і ц а 4. **Залежнасць параметраў MbT, MbP, MonP і CTP ад колькасці патокаў для алгарытмаў μ1, μ2, μ1к і μ2к, 64 радкоў у матрыцы СЛАР і 4 працэсараў**

Лік патокаў	Алгарытмы / Параметры									
	μ1 і μ1к			μ1	μ1к	μ2 і μ2к			μ2	μ2к
	MbT	MbP	MonP	CTP	CTP	MbT	MbP	MonP	CTP	CTP
4	16	16	0	0	0	31	31	0	0	0
8	8	24	16	78	2	15	31	30	974	30
16	4	28	24	238	6	7	31	30	1006	30
32	2	30	28	510	14	3	31	30	1022	30
64	1	31	30	1030	30	1	31	30	1030	30

паміж патокамі на адным працэсары застаецца сталым для μ1к, узрастае для μ2к, хутка ўзрастае для μ1 і яшчэ хутчэй для μ2.

Табл. 3, 4 паказваюць залежнасць параметраў прыкладання ад колькасці патокаў для 64 радкоў у СЛАР (4160 макра аперацый) і 4 працэсараў. Для μ2 і μ2к даўжыня крытычнага шляху не залежыць ад колькасці патокаў, пры гэтым μ2к пераўзыходзіць μ2 на 8.2%. Пры павелічэнні колькасці патокаў з 4 да 64 крытычны шлях скарачаецца, а каэфіцыент распаралельвання і загрузка працэсараў павялічваецца на 53.2% для алгарытму μ1 і на 65.8% для алгарытму μ1к. Згодна з табл. 4, лік паве-

Т а б л і ц а 5. **Залежнасць CP, Prl і Ld ад колькасці працэсараў для алгарытмаў μ1, μ2, μ1к і μ2к, 64 радкоў у матрыцы СЛАР (4160 макра аперацый) і 32 патокаў**

Лік працэсараў	Алгарытмы / Параметры											
	μ1			μ2			μ1к			μ2к		
	CP	Prl	Ld%	CP	Prl	Ld%	CP	Prl	Ld%	CP	Prl	Ld%
1	4160	1.00	100.0	4160	1.00	100.0	4160	1.00	100.0	4160	1.00	100.0
2	2176	1.91	95.6	2176	1.91	95.6	2146	1.94	96.9	2114	1.97	98.4
4	1184	3.51	87.8	1184	3.51	87.8	1142	3.64	91.1	1094	3.80	95.1
8	688	6.05	75.6	688	6.05	75.6	646	6.44	80.5	590	7.05	88.1
16	440	9.45	59.1	440	9.45	59.1	410	10.15	63.4	350	11.89	74.3
32	316	13.16	41.1	316	13.16	41.1	316	13.16	41.1	254	16.38	51.2

Т а б л і ц а 6. **Залежнасць MbT, MbP, MonP і CTP ад колькасці працэсараў для алгарытмаў μ1, μ2, μ1к і μ2к, 64 радкоў у матрыцы СЛАР і 32 патокаў**

Лік працэсараў	Алгарытмы / Параметры									
	μ1 і μ1к			μ1	μ1к	μ2 і μ2к			μ2	μ2к
	MbT	MbP	MonP	CTP	CTP	MbT	MbP	MonP	CTP	CTP
1	2	0	124	2106	62	3	0	126	4154	126
2	2	62	60	1046	30	3	63	62	2070	62
4	2	30	28	510	14	3	31	30	1022	30
8	2	14	12	230	6	3	15	14	486	14
16	2	6	4	66	2	3	7	6	194	6
32	2	2	0	0	0	3	3	0	0	0

дамленняў паміж парай патокаў памяншаецца з 16 да 1, лік паведамленняў паміж парай працэсараў павялічваецца з 16 да 31, а колькасць паведамленняў на адным працэсары павялічваецца з 0 да 30 для алгарытмаў μ1 і μ1к. Для алгарытмаў μ2 і μ2к, лік паведамленняў паміж парай патокаў памяншаецца з 31 да 1, а лік паведамленняў паміж парай працэсараў і паміж патокамі на адным працэсары нязменны і роўны 31 і 30 адпаведна. Лік перадач упраўленняў паміж патокамі на адным працэсары ўзрастае з 0 да 30 для μ1к, застаецца сталым і роўным 30 для μ2к, хутка ўзрастае з 0 да 1030 для μ1 і яшчэ хутчэй ўзрастае з 0 да 1030 для μ2.

Табл. 5, 6 паказваюць залежнасць параметраў шматпаточнага праграмнага прыкладання ад колькасці працэсараў для 64 радкоў у СЛАР і 32 патокаў. Для алгарытмаў μ1 і μ2, пры павелічэнні колькасці працэсараў з 1 да 32 даўжыня крытычнага шляху паменшылася з 4160 да 316, каэфіцыент распаралельвання вырас з 1.0 да 13.16 і загрузка працэсараў паменшылася з 100% да 41.1%. Па гэтых параметрах алгарытм μ1к даў некалькі лепшыя вынікі. Самым лепшым аказаўся алгарытм μ2к, які скарацiў даўжыню крытычнага шляху да 254 і павялічыў каэфіцыент распаралельвання да 16.38 (на 24.4%). Відавочна, што лік паведамленняў паміж парай патокаў не залежыць ад колькасці працэсараў. Дынаміка змяншэння сярэдняй колькасці паведамленняў паміж парай працэсараў і колькасці паведамленняў на адным працэсары амаль аднолькавая для ўсіх алгарытмаў, з 62–63 да 2–3 і з 124–126 да 0 адпаведна. Колькасць перадач упраўлення паміж патокамі на працэсары памяншаецца для ўсіх алгарытмаў з ростам колькасці працэсараў, але ў розным дыяпазоне. Менш за ўсё перадач у алгарытму μ1к (0–62), затым у алгарытму μ2к (0–126, каля 2 разоў больш). Алгарытмы μ1 і μ2 выконваюць нашмат больш перадач, 0–2106 і 0–4154 адпаведна.

Заключэнне

Прапанаваныя кааператыўныя блочна-паралельныя алгарытмы μ1к і μ2к значна пераўзыходзяць вядомыя алгарытмы μ1 і μ2 па даўжыні крытычнага шляху, каэфіцыенту распаралельвання, загрузцы працэсараў і перадачам кіравання паміж патокамі на працэсары. Параўнанне кааператыўных алгарытмаў μ1к і μ2к паміж сабой паказвае, што μ1к дае менш перадач кіравання паміж патокамі, аднак μ2к дае больш кароткі крытычны шлях і больш значную загрузку працэсараў.

ЛІТАРАТУРА

1. **Корнеев, В. Д.** Примеры параллельного программирования на МВС-1000 / В. Д. Корнеев // ИВМиМГ СО РАН [Электронный ресурс]: – Режим доступа: http://www2.sscc.ru/Publikacii/Primery_Prll/Primery.htm – Дата доступа: 07.04.2014.
2. **Прихожий, А. А.** Исследование методов реализации многопоточных приложений на многоядерных системах / А. А. Прихожий, О. Н. Карасик // Информатизация образования, 2014, № 1. – С. 43–62.
3. **Прихожий, А. А.** Кооперативная модель оптимизации выполнения потоков на многоядерной системе / А. А. Прихожий, О. Н. Карасик // Системный анализ и прикладная информатика, 2014, № 4. – С. 13–20.

Поступила 7.7.2015

Prihozhy A., Karasik O.

COOPERATIVE BLOCK-PARALLEL ALGORITHMS FOR TASK EXECUTION ON MULTI-CORE SYSTEM

The problem of balancing the computational load among the cores of a multi-core system and increasing the efficiency of interaction among threads in a multi-thread application is considered. The cooperative block-parallel algorithms of solving complex tasks that can be decomposed into subtasks, which decrease the number of control transfers among threads, reduce the critical path length in a parallel implementation and increase the cores load are proposed.