LMD mokslo darbai, 390–395
1998, Vilnius

Šeimininkas–darbininkas lygiagrečiojo algoritmo efektyvumo analizė

R. Čiegis (MII, VGTU), R. Šablinskas (VDU)

Nagrinėsime šeimininkas–darbininkas lygiagrečiuosius algoritmus. Juos realizuosime virtualiu lygiagrečiuoju kompiuteriu, sudarytu iš darbo stočių klasterio, sujungto lokaliu arba globaliu (INTERNET) tinklu. Skaitiniai eksperimentai atlikti PVM paketu.

1. Šeimininkas–darbininkas algoritmas

Vienas iš gerai žinomų lygiagretaus skaičiavimo algoritmų modelių yra taip vadinamas *šeimininkas–darbininkas* modelis. Šis modelis paremtas idėja, kad darbus, kurių negalima išlygiagretinti, atlieka procesas–šeimininkas, o darbus, kuriuos galima atlikti lygiagrečiai, vykdo procesai–darbininkai. Nagrinėsime uždavinių klasę, kuri tenkina sekančias sąlygas:

1. Uždavinį P galima išskaidyti į uždavinių aibę,

$$(P_j,\ j = 1, \dots, N),$$

kuri yra statiškai formuojama skaičiavimų pradžioje, arba papildoma skaičiavimo eigoje.

2. Visi uždaviniai P_j gali būti sprendžiami nepriklausomai, tačiau kiekvieno uždavinio sudėtingumas gali būti nežinomas a priori.

Toliau padarykime prielaidą, kad lygiagretusis kompiuteris yra heterogeninis, t.y. procesorių greitaeigiškumas yra nevienodas ir gali keistis skaičiavimų eigoje.
Uždavinį spręsime tokiu šeimininkas–darbininkas algoritmu:

1. *Šeimininkas* kiekvienam procesui – darbininkui iš darbų sąrašo nusiunčia po vieną uždavinį.

2. Kai šeimininkas gauna pranešimą iš darbininko, jis apdoroja rezultatą ir nusiunčia darbininkui naują užduotį.

3. Jei neišsiųstų užduočių sąrašas yra tuščias, darbininkui yra siunčiamas eilinis uždavinys, kurio dar nebaigė skaičiuoti kitas jį gavęs darbininkas.

4. *Darbininkas* gauna iš šeimininko užduotį, ją sprendžia ir nusiunčia rezultatą šeimininkui.

Pastebėsime, kad 3 žingsnis skaičiavimų pabaigoje sumažina sprendimo laiko priklausomybę nuo lėtų procesorių. Remiantis šiuo algoritmu parašytas programinis skeletas programavimo kalba C, dirbantis UNIX aplinkoje ir naudojantis paketą PVM [1]. Šio programinio skeleto pagalba yra nesunku išlygiagretinti uždavinius, tenkinančius aukščiau minėtas sąlygas – vartotojui belieka apibrėžti uždavinių sąrašą ir uždavinio sprendimo algoritmą. Toliau panagrinėsime keletą šio skeleto taikymo pavyzdžių.

2. Pirminių skaičių radimas

Sprendžiant daug skaičių teorijos, kriptografijos uždavinių yra svarbu greitai generuoti pirminių skaičių aibę. Tegul mūsų užduotis surasti visus pirminius skaičius iš intervalo $[0, M]$.

Uždavinių sąrašas

Tegul jau žinomi pirminiai skaičiai iki $\sqrt{M}$ imtinai. Tada sudarome užduočių sąrašą $(P_j,\ j = 1, 2, \ldots, K)$. Čia P_j yra užduotis rasti pirminius skaičius iš intervalo $\left[\sqrt{M} + m(j-1), \sqrt{M} + mj\right]$, kur m yra vieno intervalo ilgis $m = \left\lceil \frac{M-\sqrt{M}}{K} \right\rceil$. Norėdami minimizuoti užduočių skaičiavimo laiko persidengimą skirtinguose prosesoriuose skaičiavimų pabaigoje, užduotis numeruosime mažėjimo tvarka: $P_K, P_{K-1}, \ldots, P_1$.

Lentelė 1. Pirminių skaičių radimo uždavinys.

Klasterio greitaeigiškumas	Procesorių skaičius	Sprendimo laikas, T_p sek.	Pagreitėjimas S_p
1.00	1	3270	1.00
1.78	2	1860	1.76
2.45	3	1499	2.18
3.25	6	1123	2.91
3.60	10	995	3.29
4.20	56	882	3.71
4.50	13	747	4.38

Skaičiavimo algoritmas

Naudosime gerai žinomą Eristoteno rėčio algoritmą. Remiantis šiuo algoritmu, norėdami patikrinti, ar skaičius N yra pirminis, turime patikrinti, ar jis nesidalija iš kurio nors pirminio skaičaus, neviršijančio $\sqrt{N}$. Procesas–darbininkas skaičiuodamas savo užduotį jau turi turėti pirminių skaičių sąrašą iki $\sqrt{N}$, kurį paruošia procesas–šeimininkas.

Skaičiavimo eksperimentas

Tegul ieškome visų pirminių skaičių, mažesnių už $M = 3 \cdot 10^8$. Visą intervalą dalinamę į $m = 200$ dalių, uždavinius P_j darbininkai skaičiuoja nepriklausomai. Lentelėje 1 pateikti skaičiavimo rezultatai skirtingo greitaeigiškumo virtualiesiems kompiuteriams. Čia greitaeigiškumo vienetu paimtas kompiuteris 2xPentium 300 MHz.

Lenteléje $S_p = T_1/T_p$ yra pagreitėjimas, T_1 yra laikas, per kurį užduotį atlieka kompiuteris su sąlyginiu greičiu 1, o T_p – laikas, per kurį užduotį suskaičiuoja virtualusis kompiuteris su sąlyginiu pajėgumu p. Iš lentelės rezultatų matome, kad greičiausiu kompiuteriui uždavinį išsprendžiame per 55 minutes, kai tuo tarpu 13 procesorių virtualusis kompiuteris darbą atlieka daugiau negu keturis kartus greičiau – per 12 minučių.

3. Lazerinės optikos uždavinys

Sprendžiant kai kuriuos lazerinės fizikos eksperimento modeliavimo uždavinius, atsiranda didelis skaičiavimo resursų poreikis. Paimsime pavyzdžiu lazerio spindulio projekcijos skaičiavimo uždavinį, kuriame šviesos šaltinis, sklindantis iš srities A, apšviečia plokštelę B. Taške $P = (x, y)$ apšviestumas $u(P)$ yra integralas pagal šviesos šaltinio plokštumą:

$$u(P) = \frac{1}{i\lambda} \iint_A \exp - \left(\frac{x'^2}{{w_x}^2} + \frac{y'^2}{{w_y}^2} \right) \frac{e^{iks} \cdot z}{s^2} \, dx' dy' , \tag{1}$$

$$s = \sqrt{(x - x')^2 + (y - y')^2 + z^2}, \tag{2}$$

čia w_x ir w_y – lazerio spindulio pločio parametrai. Norint gauti lazerio spindulio vaizdą plokštelėje B, reikia apskaičiuoti šviesos intensyvumą $|u(x_i, y_j)|^2$, kur (x_i, y_j) yra pasirinkti taškai.

Uždavinių sąrašas

Skaičiuokime lazerio bangos intensyvumą $M \times N$ taškų plokštelėje B. Tada darbų sąrašas yra aibė taškų $\{(x_i, y_j); 1 \leqslant i \leqslant M, 1 \leqslant i \leqslant N\}$, kuriuose apšviestumą galima skaičiuoti nepriklausomai. Uždavinių sąrašas dar padidėja, jei atskirai skaičiuojame realiąją ir menamąją šviesos intensyvumo dalis.

Skaičiavimo algoritmas

Kiekviename taške (x_i, y_j) reikia apskaičiuoti greitai osciliuojančios funkcijos integralą. Skaitinį integralo artinį rasime Genzo–Maliko adaptyviuoju algoritmu [2].

Skaičiavimo eksperimentas

Pasirinksime projekcijos taškų skaičių $M = 8$ ir $N = 8$. Pareikalaukime skaičiavimo tikslumo $\varepsilon = 0.01$.

Iš Lentelėje 2 pateiktų rezultatų matome, kad skaičiuojant su nedideliu procesorių skaičiumi, yra pasiekiamas maksimalus efektyvumas. Taip yra dėl to, kad atskiri uždaviniai yra gana sudėtingi ir komunikacijos tarp procesų laikai lyginant su skaičiavimo laiku yra maži. Įtraukiant į klasterį vis labiau nutolusius kompiuterius, komunikacijos laikai auga, todėl bendras pagreitėjimas yra mažesnis už kompiuterių klasterio galingumą.

Lentelė 2. Lazerinės optikos uždavinio sprendimas, kai $M = 8$, $N = 8$.

Klasterio greitaeigiškumas	Procesorių skaičius	Sprendimo laikas, T_p sek.	Pagreitėjimas S_p
1.00	1	2108	1.00
1.82	2	1162	1.81
2.30	3	941	2.24
3.03	4	741	2.84
3.81	10	645	3.27

4. Daugiamačių integralų skaičiavimo uždavinys

Ieškosime daugiamačio integralo

$$(f, \Omega) = \int_{\Omega} f(x)\, dx \tag{3}$$

skaitinio artinio duotu tikslumu ε, kur $\Omega = [a_1, b_1] \times [a_2, b_2] \cdots [a_n, b_n]$ yra integravimo rėžiai, o $f(x)$ yra pointegralinė funkcija. Daugiamačių integralų skaitinės aproksimacijos uždaviniai yra imlūs skaičiavimams (žr. [3]), ypač kai srities Ω dimensijos skaičius yra didelis.

Uždavinių sąrašas

Suskaidysime visą integravimo sritį į N hipersričių: $\Omega = \bigcup_{i=1}^{N} \Omega_i$. Integralų $I(f, \Omega_i)$ aibė ir bus uždavinių sąrašas, kurį galima spręsti lygiagrečiai. Reikalausime, kad integravimo tikslumas kiekvienoje iš hipersričių būtų ε/N, t.y. paklaidą paskirstome tolygiai visoje integravimo srityje.

Uždavinio skaičiavimo algoritmas

Gautus uždavinius $I(f, \Omega_i)$ skaičiuosime adaptyviu Genzo–Maliko algoritmu (žr. [2]).

Skaičiavimo eksperimentas

Skaičiuosime aštuonmatį integralą

$$\int_0^1 \int_0^1 \cdots \int_0^1 \sum_{i=1}^{8} \exp(2x_i)\, dx\ , \quad \varepsilon = 10^{-6}. \tag{4}$$

Pasirinksime $N = 256$. Lentelėje 3 pateikti skaičiavimų rezultatai įvairaus galingumo kompiuterių klasteriams.

Lentelė 3. Daugiamačių integralų skaičiavimo uždavinys.

Klasterio greitaeigiškumas	Procesorių skaičius	Sprendimo laikas, T_p sek.	Pagreitėjimas S_p
1.00	1	2105	1.00
2.10	3	1006	2.09
4.00	11	539	3.91
5.10	13	435	4.84

Lentelė 4. Dvimačio integralo skaičiavimo uždavinio rezultatai.

Užduočių skaičius N	Padalinimų skaičius Q_N	Kokybė $q = Q_1/Q_N$
1	88684	1.00
4	137421	0.65
16	209707	0.42
64	304515	0.29

Skaičiuojant integralus, kuriuose pointegralinė funkcija turi singuliarumo taškų, toks lygiagretus algoritmas nėra efektyvus dėl dviejų priežasčių. Apibrėžkime integravimo algoritmo sudėtingumą Q kaip Genzo–Maliko algoritmo padalinimų, reikalingų norint apskaičiuoti integralo artinį ε tikslumu, skaičių. Tada lygiagretaus algoritmo sudėtingumas yra didesnis už geriausiojo nuoseklaus algoritmo sudėtingumą ir, didinant užduočių skaičių N, jis didėja. Lentelėje 4 pateiktuose rezultatuose yra skaičiuojamas integralas

$$\int_0^1 \int_0^1 (x_1 x_2)^{-0.95}\, dx_1\, dx_2 \quad , \ \varepsilon = 10^{-2}. \tag{5}$$

Iš lentelės matome, kad didinant užduočių skaičių, algoritmo kokybė mažėja, t.y. atliekami pertekliniai skaičiavimai. Taip atsitinka todėl, kad paklaidą paskirstome tolygiai visoms hipersritims, o Genzo–Maliko algoritmas šią paklaidą paskirsto adaptyviai. Antroji mažo lygiagretaus algoritmo efektyvumo priežastis yra ta, kad procesas darbininkas, gavęs hipersritį su tašku $(0, 0)$, atlieka didžiąją darbo dalį. Jei pradinis darbų skaičius yra 16, tai procesas, gavęs hipersritį su singuliarumu $(0, 0)$, smulkins jį 208096 kartų, kai tuo tarpu visos kitos užduotys kartu smulkinamos tik 1610 kartų. Tam, kad lygiagretus algoritmas būtų efektyvus integralams su singuliariomis pointegralinėmis funkcijomis, reikia modifikuoti užduočių formavimo algoritmą.

LITERATŪRA

[1] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM: Parallel Virtual Machine*, MIT Press, Cambridge, Massachusetts, London, 1993.

[2] A. C. Genz and A. A. Malik, Remarks on Algorithm 006: an adaptive algorithm for numerical integration over an N-dimensional rectangular region, *J. Comput. Appl. Math.*, **6** (1980), 295–302.

[3] A. H. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice Hall, Englewood Cliffs, New Jersey, 1971.

The analysis of the efficiency of a master–slave parallel algorithm

R. Čiegis, R. Šablinskas

A general master–slave parallel algorithm is described. Three applications are investigated and results of numerical experiments with various clusters of workstations are given. PVM library is used in computations as a message passing interface.