Проект

по Системи за Паралелна Обработка

Паралелно обхождане в широчина на граф (BFS)

Изготвил:

Станимир Николаев Николов

Фн.80397 КН

Ръководител:

ас. Христо Христов

Проверил: ........................

(ас. Христо Христов)

*1. Постановка на задачата и подход за решаването й*

Разглеждаме графа G(V,E). Графа e неориентиран, свързан или

слабо свързан. За представянето на графа ще използваме матрица на съседство. Да се напише програма, която реализира обхождане на графа G в ширина. Използваме следната схема:

1. Като начална стъпка на обхождането имаме един връх. От него си образуваме множество от върхове, нека го наречем ниво 0. Отбелязваме го за обходен.
2. Намираме се на ниво К. Образуваме множеството ниво К+1 като това са всички необходими съседи на върхове от ниво К. При добавянето им ги отбелязваме за обходени.
3. Връщаме се на стъпка 1, ако левел К + 1 не е празното множество.

Тази схема се повтаря за всеки един от върховете на графа, който не е обходен. Накрая сме сигурни, че сме обходили всички върхове, без значение дали графа е свързан или не.

Реализираме точка 2 от горе описаната схема многонишково, като разделяме елементите на множеството К на T-наброй подмножества, като Т е броят на нишките. Така за всяка нишка ще има по почти еднакъв брои върхове които да обходи. (Последното подмножество ще съдържа броя на елементите на К / Т + К mod (T) )

Всяка от подзадачитеще е средно и също натоварване, тоест с една и съща времева сложност, която при това не зависи от специфичните особености на входните данни. Това позволява да се постигне максимално балансиране на натоварването върху отделните изчислителни единици.

*2. Описание на реализирания алгоритъм*

За паралелното решаване на задачата за обхождане на граф в широчина е избран подходът разделяй и владей, който е реализиран чрез многонишково програмиране. Той предоставя възможност за оптимално намаляване на използвания ресурс -време, тъй като позволява отделните процеси извършват изчисления едновременно използвайки еднии същи входни данни и да пресмятат частични крайни резултати, които чрез сумиране да дадат желания резултат.Алгоритъмът е реализиран със синхронизация, тъй като се налага отделните подпроцеси да пишат на общо място в паметта (опашката next) и работят със споделени функции(AtomicBitset).

Алгоритъмът е аналогичен на обикновено BFS. Само че вместо да имаме една опашка, от която да взимаме по един връх от началото и за всичките негови съседи, който не са обходени, да ги добавяме на края на опашката, си имаме две опашки current и next. В current си държим много върхове, а не както при стандарното решение само 1. Целта е да образуваме next – това са върховете от следващото ниво. Това ще бъде реализарано благодарение на няколко на брои slave нишки. Освен това искаме да се постигне оптимално разделяне на данните, които всяка нишка ще обработи, така ще се ускори изчислителния процес като позволи оптимално използване на ядрата на процесора. На всяка нишка ще се падне по (броя на елементите в current)/(броя на нишките), освен на последната. Тя ще трябва да обходи малко повече върхове по (броя на елементите в current)/(броя на нишките) + по (броя на елементите в current)mod(броя на нишките). Когато нишка приключи с обхождането на всички върхове в своята част на current, тя синхронизирано добавя своят резултат, който е бил записал локално за нея в next. След това нишката приключва работа.

Когато всички slave нишки приключат работа, тогава в опашката next вече са всички върхове от следващото ниво.

Остава само Master нишката която прави Current да е вече Next и повтаря изброениете стъпки докато не обходим целят граф.

След приключване на алгоритама програмата извежа колко време е отнело. И ако не е подаден входен параметър –q какъв е резултатът:

Масив Levels – в който за всеки връх от 0 до n е записано цяло число – на кой левел се намира даденият връх

Масив Parents – в който за всеки връх от 0 до n e записано цяло число – номера на баща му.

Освен това ако не е подаден параметър –q програмата извежда и съобщения с цел информиране на текущата деиност на програмата.

Входните данни са:

-n брой върхове

-q (това е тих режим в който излиза само времето за което е изпълнена задачата)

-t брой нишки (брой нишки които ще извършват изчисленията)

*3. Тестови замервания и анализ на метричните показатели*

Разработено е приложение на Java и е тествано на 2 ядрен процесор (core 2 duo Celeron E3300) и на четириядрен (Phenom X4 945) с цел да се оценят ускорението S (забързване, speed-up) и ефективността Е(efficiency) на описания алгоритъм, където ако T(p) е времето необходимо за завършване на работата на алгоритъм с p на брой нишки то:

S(p) = T(1)/T(p)

E(p) = S(p)/p

Замерванията направени за 10 000 върха.

Celeron E3300

|  |  |  |  |
| --- | --- | --- | --- |
| **Нижки** | **Време за изпълнение(сек)** | **Ускорение** | **Ефективност** |
| 1 | 1,94502661 | 1 | 1 |
| 2 | 1,163366295 | 1,671895274 | 0,835947637 |
| 3 | 1,326295879 | 1,466510332 | 0,488836777 |
| 4 | 1,124236317 | 1,730086976 | 0,432521744 |
| 5 | 0,889371549 | 2,186967429 | 0,437393486 |
| 6 | 1,04262609 | 1,865507327 | 0,310917888 |
| 7 | 0,96680074 | 2,011817461 | 0,287402494 |
| 8 | 0,932165823 | 2,086567177 | 0,260820897 |

Phenom X4 945

|  |  |  |  |
| --- | --- | --- | --- |
| **Нижки** | **Време за изпълнение(сек)** | **Ускорение** | **Ефективност** |
| 1 | 1,105800318 | 1 | 1 |
| 2 | 0,450292425 | 2,455738219 | 1,22786911 |
| 3 | 0,298773169 | 3,701136624 | 1,233712208 |
| 4 | 0,230454429 | 4,798347 | 1,19958675 |
| 5 | 0,295539517 | 3,74163269 | 0,748326538 |
| 6 | 0,240538113 | 4,597193784 | 0,766198964 |
| 7 | 0,257381036 | 4,296355066 | 0,613765009 |
| 8 | 0,238101211 | 4,644244829 | 0,580530604 |