Παράλληλη εφαρμογή Της Scan με OpenMP

Νικόλαος Αυγούστης

23 Ιανουαρίου 2020

Περιεχόμενα

1 Λειτουργιά Scan

Η λειτουργία Scan είναι να περνάει πάνω από μια ακολουθία στοιχείων $x_0, x_1, x_2, ...$ και παράγει μια δεύτερη ακολουθία $y_0, y_1, y_2, ...$ η οποία παριστά τα επιμέρους αθροίσματα των στοιχείων της πρώτης βάσης της παρακάτω λογικής

$$y_0 = x_0$$
$$y_1 = x_0 + x_1$$
$$y_2 = x_0 + x_1 + x_2$$
$$...$$

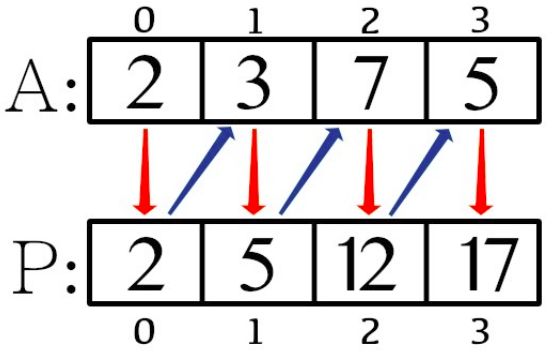
Ένα απλό παράδειγμα της λειτουργίας είναι δοθέντος της ακολουθίας

x_i	1	2	3	4	...

Παίρνω τα παρακάτω επιμέρους αθροίσματα

y_i	1	3	6	10	...

Η σειριακή υλοποίηση της Scan είναι απλοϊκή με τη χρήση ενός επαναληπτικού βρόγχου ξεκινώντας από το δεύτερο στοιχείο και κάθε φορά υπολογίζοντας το κάθε στοιχείο με τον εξής τύπο $y_i = y_{i-1} + x_i$.



(α′) Γραφική αναπαράσταση της σειριακής Scan .

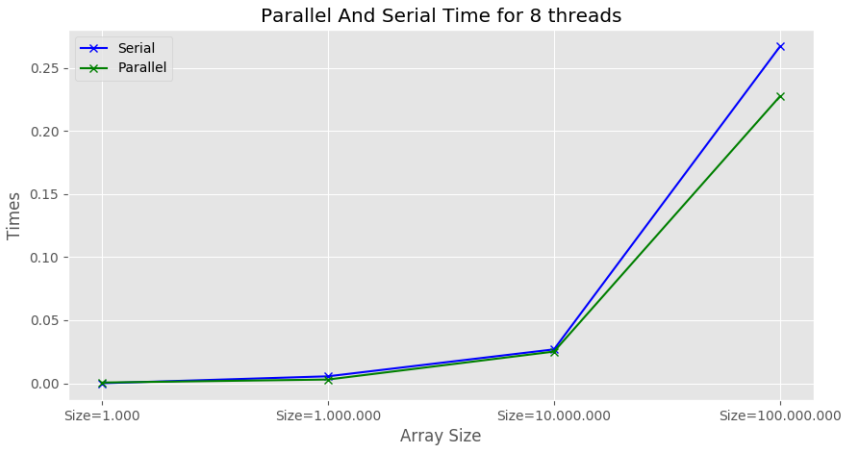
```
int *a;
a = (int *)malloc(SIZE*sizeof(int));
for(int i=1;i<SIZE;i++){
        a[i] = a[i-1] + a[i];
}
```

(β′) Σκιαγράφηση κώδικα σε γλώσσα C της σειριακής Scan .

2 Παράλληλη Scan

2.1 Αλγόριθμος κοινής μνήμης

Ο αλγόριθμος που υλοποίησα στα πλαίσια της εργασίας υποθέτει μηχανή κοινής μνήμης (shared memory) όπου όλα τα επεξεργαστικά στοιχεία έχουν πρόσβαση σε αυτήν.
Για τον υπολογισμό των επιμέρους αθροισμάτων n στοιχείων από p επεξεργαστικά στοιχεία παράλληλα, τα δεδομένα μοιράζονται σε $p+1$ τμήματα, τα οποία περιέχουν το καθένα $\frac{n}{p+1}$ δεδομένα. Ο αλγόριθμος αποτελείτε από 3 τμήματα.
Στην πρώτη σάρωση κάθε επεξεργαστικό στοιχείο υπολογίζει ένα τοπικό άθροισμα για το τμήμα του. Το τελευταίο τμήμα δεν χρειάζεται να υπολογιστεί γιατί τα αθροίσματα υπολογίζονται βάση των προθεμάτων τον προηγουμένων τμημάτων και το τελευταίο δεν έχει κάποιο τμήμα να το διαδέχεται.
Στο δεύτερο τμήμα τα p αυτά αθροίσματα αποθηκεύονται στο τελευταίο στοιχείο κάθε τμήματος και μαζεύονται και αυτά σε ένα δικό τους επιμέρους άθροισμα και αποθηκεύονται στις επόμενες θέσεις τους για να χρησιμοποιηθούν στο τρίτο μέρος.
Τέλος στο τρίτο μέρος κάθε επεξεργαστικό στοιχείο σαρώνει άλλη μια φορά αυτή τη φορά δεν χρειάζεται ο υπολογισμός του πρώτου τμήματος, επειδή δεν έχει προηγούμενο τμήμα ώστε να έχω άθροισμα το οποίο να πρέπει να υπολογιστεί, αυτή τη φορά όμως υπολογίζουμε το τελευταίο τμήμα και τα αθροίσματα υπολογίζονται λαμβάνοντας υπόψη και τα ′μετατοπισμένα αθροίσματα′ που υπολογίστηκαν στο προηγούμενο βήμα.



2.2 Κώδικας σε C

Παρακάτω παρατίθεται ο κώδικας σε γλώσσα C που υλοποιήθηκε.

```
#include <stdio.h>
#include "omp.h"
#include <stdlib.h>

#define SIZE 1000000000
#define NUMTHREADS 8
int main(){
unsigned long int *data,*p_sum;
data = (unsigned long int *)malloc(SIZE*sizeof(unsigned long int));
const int nthreads = NUMTHREADS;
double time_spent = 0.0;
for(unsigned long int i=0; i<SIZE; i++){
        data[i] = i+1;
}
p_sum=(unsigned long int *)malloc((nthreads+1)*sizeof(unsigned long int));
for(int i=0;i<nthreads+1;i++){
    p_sum[i]=0;
}
double begin = omp_get_wtime();
#pragma omp parallel num_threads(NUMTHREADS)
{
    const int tid = omp_get_thread_num();
    unsigned long int  sum = 0;
    #pragma omp for schedule(static)
    for (unsigned long int i=0; i<SIZE; i++) {
        sum += data[i];
        data[i] = sum;
    }
    p_sum[tid+1] = sum;
    #pragma omp barrier
```

```
        unsigned long int offset = 0;
        for(int i=0; i<(tid+1); i++) {
            offset += p_sum[i];
        }
        #pragma omp for schedule(static)
        for (unsigned long int i=0; i<SIZE; i++) {
            data[i] += offset;
        }
    }
    double end = omp_get_wtime();
    time_spent = (double)(end - begin);
    printf("%f\n", time_spent);
    free(data); free(p_sum);
    return 0;
    }
```

2.3 Ανάλυση Work/Span του αλγορίθμου

2.3.1 Work

Ο αλγόριθμος αυτός χωρίζεται σε τρία τμήματα από τα οποία το πρώτο και το τρίτο εξαρτώνται από των αριθμό των στοιχείων μας προς αριθμό των επεξεργαστικών στοιχείων άρα οι χρόνοι των δυο αυτών θα είναι $O(\frac{n}{p})$ και το δεύτερο αν γίνει παράλληλα εξαρτάται μόνο από το πλήθος τον επεξεργαστικών μας στοιχείων άρα $O(p)$ οπότε σε ένα σύστημα με συγκεκριμένο αριθμό επεξεργαστικών στοιχείων αυτό θεωρείτε σταθερό και το μόνο που αλλάζει είναι το πλήθος των στοιχείων μας άρα τελικά το Work συνολικά θα είναι $O(\frac{n}{p}) + O(p) + O(\frac{n}{p}) = O(n)$.

2.3.2 Span

Για το Span του αλγορίθμου πρέπει να λάβουμε υπόψη ότι ανάλογα το πλήθος των δεδομένων μας και των επεξεργαστικών στοιχείων μας χωρίζονται σε $p+1$ τμήματα όπου στο πρώτο μέρος υπολογίζονται παράλληλα p από αυτά διότι δεν χρησιμοποιείτε το τελευταίο και όμοια στο τρίτο μέρος όπου δεν χρησιμοποιούμε το πρώτο μέρος και το μεγαλύτερο μονοπάτι έχει μήκος $\frac{n}{p}$. Στο δεύτερο μέρος τώρα υπολογίζονται είτε σειριακά είτε παράλληλα $p-1$ αθροίσματα οπότε τελικά έχω $\frac{n}{p} + (p-1) + \frac{n}{p} = O(n)$.

2.4 Αλγόριθμος των Hillis & Steele

Ο αλγόριθμος που παρουσιάζουν οι Hillis & Steele αποτελεί έναν αρκετά παραλληλισιμο αλγόριθμο με μικρότερο Span καθώς για n επεξεργαστικά στοιχεία απαιτεί $log_2(n)$ επανάληψης αλλά δεν είναι αποδοτικός διότι εκτελεί $log_2(n)$ πράξεις για την επιτυχία του αποτελέσματος ενώ ο σειριακός αλγόριθμος εκτελεί n πράξεις.

2.5 Αλγόριθμος κατανεμημένης μνήμης

Ο αλγόριθμος υπέρ κύβου είναι ένας αλγόριθμος υπολογισμού των αθροισμάτων προσαρμοσμένος για συστήματα κατανεμημένης μνήμης, λειτουργεί με την ανταλλαγή μηνυμάτων μεταξύ των στοιχείων επεξεργασίας. Υποθέτει ότι έχουμε $p = 2^d$

επεξεργαστικά στοιχεία που συμμετέχουν στον αλγόριθμο ίσο με τον αριθμό των γωνιών σε ένα $d-$διαστατό υπέρ κύβο.
Σε όλο τον αλγόριθμο, κάθε επεξεργαστικό στοιχείο θεωρείται γωνία ενός υποθετικού υπέρ κύβου με γνώση του συνολικού ποσού του επιμέρους αθροίσματος σ καθώς και το επιμέρους άθροισμα x όλων των στοιχείων μέχρι το δικό του υπέρ κύβου.
Ο αλγόριθμος ξεκινάει υποθέτοντας ότι κάθε επεξεργαστικό στοιχείο είναι η μοναδική γωνία ενός υπέρ κύβου με μηδενικές διαστάσεις και ως εκ τούτου σ και x είναι ίσες με το τοπικό άθροισμα προθέματος των δικών του στοιχείων.
Ο αλγόριθμος συνεχίζεται με την ενοποίηση των υπέρ κύβων που είναι δίπλα κατά μια διάσταση. Κατά τη διάρκεια κάθε ενοποίησης, το σ ανταλλάσσεται και συσσωρεύεται μεταξύ των δύο υπέρ κύβων και όλα τα επεξεργαστικά στοιχεία στις γωνίες αυτού του νέου υπέρ κύβου αποθηκεύουν το άθροισμα.
Ωστόσο, μόνο ο υπέρ κύβος που περιέχει τα επεξεργαστικά στοιχεία με υψηλότερο δείκτη προσθέτει και αυτή την σ στην τοπική τους μεταβλητή ξ διατηρώντας αμετάβλητο το x του επιμέρους αθροίσματος όλων των στοιχείων σε επεξεργαστικά στοιχεία με δείκτες μικρότερους ή ίσους με το δικό τους δείκτη.
Σε έναν $d-$διαστατό υπέρ κύβο με 2^d επεξεργαστικά στοιχεία στις γωνίες του ο αλγόριθμος πρέπει να επαναληφθεί d φορές για να έχει ενοποιήσεις τους 2^d μηδενικής διάστασης υπέρ κύβους σε έναν $d-$διαστατό υπέρ κύβο.