Αναφορά 2ης εργασίας για το μάθημα Αρχιτεκτονική Παράλληλων και Κατανεμημένων Υπολογιστών

2 Ιουνίου 2019

Κωνσταντίνος Κασφίκης – 2013030108
Γεώργιος Λουκόπουλος – 2013030133

Στην συγκερκιμένη εργαστηριακή άσκηση ζητούνταν η τροποποίηση μιας απλοποιημένης μορφής υπολογισμού του omega statistic με την χρήση SSE,Pthreads και MPI.

SSE:
Με την χρήση SSE (Streaming SIMD Extensions) εισάγεται ο τύπος μεταβλητής **__m128**. Μια τέτοια μεταβλητή είναι σε θέση να αποθηκεύσει 4 floats , επιτρέποντας την διεκπεραίωση πράξεων σε ομάδες των 4 float. Κατά αυτόν τον τρόπο, αντί να πραγματοποιούμε πράξεις μεταξύ των πινάκων ένα στοιχείο την φορά, έχουμε την δυνατότητα να διασχίζουμε τους πίνακες ανά 4 στοιχεία, υπολογίζοντας έτσι 4 τιμές του FVec. την φορά.
Συγκεκριμένα, αρχικά φορτώνουμε τις αντίστοιχες τετράδες απο τους πίνακες μέσω της _mm_load_ps(). Έπειτα, κάνουμε τους κατάλληλους υπολογισμούς για τις τετράδες βάση των εξισώσεων της εκφώνησης χρησιμοποιώντας τις εντολές _mm_add_ps(), _mm_sub_ps(), _mm_mul_ps() και _mm_div_ps() και αποθηκεύουμε την τετράδα των αποτελεσμάτων στον πίνακα FVec. Εφόσον το __m128 F_result περιέχει 4 στοιχεία του πίνακα,υπάρχει ανάγκη εύρεσης του μεγαλύτερου και μικρότερου στοιχείου.Με την χρήση 2 φορές των εντολών _mm_min_ps(), _mm_max_ps() επιτυγχάνουμε τη σύγκριση όλων των float μέσα στο __m128 F_result με όλους και την αντικατάσταση όλων αυτών με

τη μικρότερη/μεγαλύτερη τιμή. Η τιμή που θα προκύψει θα αποθηκευτεί στο minf/maxf. αντίστοιχα.
Όσον αφορά το Avgf. πρέπει να προσθέσουμε το σύνολο των 4 float που εμπεριέχονται μεσα στο __m128 F_result και να αντικαταστίσουμε ολα τα float με αυτα με την τιμη του αθροίσματος. Για να επιτύχουμε τα παραπάνω χρησιμοποιούμε διαδοχικά 2 φορες την εντολή _mm_hadd_ps(). Γενικά στις εντολές SSE η κατάληξη __ss δεικνύει οτι θα χρησιμοποιηθεί μόνο το πρώτο float από την τετράδα, ενώ στο __ps χρησιμοποιούντε και τα 4. Επειδή το πρόγραμμα μας πρέπει να είναι σε θέση να επεξεργαστεί οποιονδήποτε N αριθμό στοιχείων, πρέπει να λάβουμε υπόψιν μας τις περιπτώσεις όπου το N%4 != 0, επεξεργαζόμαστε λοιπόν τις διαθέσιμες τετράδες με __ps εντόλες, ενώ τα εναπομείναντα στοιχεία επεξεργάζονται με την χρήση __ss εντολών(ένα στοιχείο την φορά)

SSE+Pthreads:
Στόχος του μέρους αυτού του εργαστηρίου είναι η παραλληλοποίηση του κώδικα SSE με τη χρήση pthreads Ο χρήστης ορίζει έναν αριθμό από threads, στα οποία μαζί με το main thread θα μοιραστεί ο φόρτος εργασίας.
Αρχικά, δημιουργήθηκε η συνάρτηση omega_calc(), η οποία περιέχει το σύνολο των απαιτούμενων SSE εντολών (που υλοποιήθηκαν στο προηγούμενο μέρος) για τον υπολογισμό του omega statistic. Εφόσον η omega_calc(), θα είναι η συνάρτηση που θα εκτελέσει το κάθε thread δέχεται ορίσματα και επιστρέφει τιμή τύπου (void*) , οπότε είναι απαραίτη η δημιουργία ενός struct που προορίζεται για την μεταφορά των ορισμάτων στην συνάρτηση και αντίστοιχα ενός προκειμένου να μπορούμε να αξιοποιήσουμε τις τιμές που επιστρέφονται. Εκτός απο την παραπάνω συνάρτηση, απαραίτητη ήταν και η δημιουργία της spawn_threads() , συνάρτηση η οποία είναι υπεύθυνη για την αρχικοποίηση των threads. , καθώς και για τον διαμοιρασμό φόρτου εργασίας σε κάθε thread ,συμπεριλαμβανομένου και του main. Η spawn_threads() είναι υπεύθυνη για την δέσμευση χώρου και αρχικοποίηση των struct ορισμάτων του κάθε thread , βάσει του οποίου θα εκτελεστεί η εκάστοτε omega_calc(). Στο struct ορισμάτων μεταξύ άλλων εμπεριέχονται 2 integers (from_index,to_index) τα οποία ορίζουν για ποια N θα τρέξει το κάθε thread (πχ για N= 1000 και για num_of_threads = 4, το main thread θα επεξεργαστεί τα δεδομένα απο 0-199,το 1ο από 200-399,το 2ο απο 400-599,το 3ο απο 600-799 και το 4ο απο 800-999).
Η spawn_threads() καλείται πρώτη φορά στην αρχή της main (αμέσως μετά από τη δέσμευση των απαραίτητων πινάκων) και δημιουργεί threads τα οποία θα υπολογίσουν το omega statistic για το πρώτο iteration Κατά την αρχικοποίηση των

δεδομένων, τα threads που δημιουργήθηκαν πρέπει να βρίσκονται σε κατάσταση busy-wait , δηλαδή να αναμένουν χωρίς να εκτελούν κάποια λειτουργία. Μετά την αρχικοποίση των δεδομένων μπορούν να ξεκινήσουν να επεξεργάζονται τα δεδομένα που τους ανατέθηκαν. Για την υλοποίηση της busy-wait λειτουργίας χρησιμοποιήθηκε condition variable στο οποίο τα threads αναμένουν (cond_wait) εφόσον πληρείται μια συνθήκη. Το main thread μόλις ολοκληρώσει την αρχικοποίηση των πινάκων σηματοδοτεί (cond_broadcast()) όλα τα τηρεαδς που αναμένουν στο συγκεκριμένο condition variable να συνεχίσουν την λειτουργία τους. Πλέον, εφόσον σε κάθε iteration χρησιμοποιούνται τα ίδια δεδομένα (γίνεται μόνο μια φορά αρχικοποίηση των πινάκων), δεν υπάρχει η ανάγκη τα threads που δημιουργούνται για τα iteration έπειτα του πρώτου να περιμένουν στο condition variable, οπότε πλέον δεν πληρείται η συνθήκη που προαναφέρθηκε και τα threads δεν αναμένουν στο condition variable από το δεύτερο iteration και μετά, αλλά ξεκινούν κατευθείαν την λειτουργία τους. Σε κάθε iteration υπάρχει η ανάγκη αναμονής του main thread (εφόσον έχει τελειώσει την επεξεργασία των δεδομένων που του ανατέθηκαν) μέχρι όλα τα υπόλοιπα να τελειώσουν επίσης την λειτουργία τους. Χρησιμοποιείται,λοιπόν,για κάθε thread η thread_join() , μέσω της οποίας μπορούμε να έχουμε πρόσβαση στα struct επιστροφής του κάθε thread.

SSE+Pthreads+MPI:
Με την χρήση MPI, δίνεται η δυνατότητα δημιουργίας πολλαπλών διεργασιών, οι οποίες έχουν την δυνατότητα επικοινωνίας μεταξύ τους, όπου κάθε διεργασία θα επεξεργαστεί μέρος τους συνολικού φόρτου εργασίας, το οποίο έπειτα θα διαμοιραστεί στα thread της κάθε διεργασίας. Για την ορθή λειτουργία των διεργασιών, καθορίζεται μια διεργασία που λειτουργεί ως master και είναι υπεύθυνη για τον συντονισμό των υπόλοιπων worker διεργασιών, χωρίς ωστόσο αυτό να σημαίνει ότι η master διεργασία δεν επεξεργάζεται και αυτή μέρος του φόρτου εργασίας. Κάθε διεργασία δεσμεύει και αρχικοποιεί του απαραίτητους πίνακες ανεξάρτητα, ωστόσο στο τέλος της λειτουργίας τους υπάρχει η ανάγκη μεταφοράς των μερικών αποτελεσμάτων τους (maxF,minF,avgF καθώς και το σύνολο των στοιχείων Fvec που υπολόγισαν) στην master διεργασία για την εύρεση των συνολικών αποτελεσμάτων. Άρχικα, εφόσον πλέον υπάρχουν P διεργασίες (ορισμένες από τον χρήστη), κάθε διεργασία θα επεξεργαστεί N/P περίπου στοιχεία,τα οποία θα ανατεθούν σε κάθε διεργασία απο την master. Στην αρχή εκτέλεσης της κάθε διεργασίας γίνονται τα εξής:
Master:
υπολογίζει το πως θα μοιραστεί ο φόρτος εργασίας, και στέλνει σε όλες τις

υπόλοιπες διεργασίες 2 integer μέσω της εντολής MPI_send τα οποία ορίζουν ποια στοιχεία θα επεξεργαστεί η κάθε διεργασία. Βάσει αυτών των integer διαμοιράζεται πλέον ο φόρτος εργασίας της κάθε διεργασίας στα threads του.
Worker:
μέσω της εντολής MPI_Recv δέχεται τις προαναφερθέντες μεταβλητές.
Στην συνέχεια, ακολουθεί η αρχικοποίηση των πινάκων κάθε διεργασίας, διαδικασία κατά την οποία τα threads της κάθε διεργασίας βρίσκονται σε κατάσταση busy-wait. Μόλις αρχικοποιηθούν τα δεδομένα, τα threads αρχίζουν να λειτουργούν. Μόλις η λειτουργία των threads ολοκληρωθεί γίνονται τα εξής:
Workers:
Μέσω της MPI_send οι workers στέλνουν τα μερικά τους αποτελέσματα στην master διεργασία.
Masters:
Μέσω της MPI_Recv η master δέχεται τα μερικά αποτελέσματα από όλους τους workers, τα συνδυάζει και τελικά εκτυπώνει τα αποτελέσματα.

Σύγκριση αποτελεσμάτων:

```
^Cmd@MDpc:~/Desktop/project$ ./run.sh

The run script for the project
Executing the refference code for N=10,000,000
Omega time 0.232895s - Total time 1.474977s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.529043e-01
finished..
Executing the SSE inplementation for N=10,000,000
Omega time 0.037043s - Total time 1.396129s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.317009e-01
finished..
Executing the SSE and pthreads inplementation for num_of_threrads = 2
Omega time 0.028909s - Total time 2.067895s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.319634e-01
finished..
Executing the SSE and pthreads inplementation for num_of_threrads = 4
Omega time 0.026222s - Total time 1.993605s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.319637e-01
finished..
Executing the SSE and pthreads and MPI inplementation for num_of_threrads = 2 and num_of_procs = 2

LAM 7.1.4/MPI 2 C++/ROMIO - Indiana University

Omega time 0.030820s - Total time 2.045300s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.319595e-01
finished..
Executing the SSE and pthreads and MPI inplementation for num_of_threrads = 2 and num_of_procs = 4

LAM 7.1.4/MPI 2 C++/ROMIO - Indiana University

Omega time 0.122309s - Total time 5.503323s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.319628e-01
finished..
Executing the SSE and pthreads and MPI inplementation for num_of_threrads = 4 and num_of_procs = 2

LAM 7.1.4/MPI 2 C++/ROMIO - Indiana University

Omega time 0.023462s - Total time 1.985897s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.319596e-01
finished..
Executing the SSE and pthreads and MPI inplementation for num_of_threrads = 4 and num_of_procs = 4

LAM 7.1.4/MPI 2 C++/ROMIO - Indiana University

Omega time 0.075926s - Total time 5.651974s - Min -2.711918e+06 - Max 6.048419e+05 - Avg -6.319594e-01
finished..
```

Στην παραπάνω εκτέλεση του run.sh παρατηρούμε ότι ο δοθέν κώδικας ολοκληρώνει την διαδικασία omega statistics (χωρίς αρχικοποίηση) αισθητά πιο αργά από τον αντίστοιχο κώδικα με SSE. (Omega Time Orig – 0.23 ενώ Omega

Time SSE – 0.03). Αντίστοιχα η παραλληλοποίηση του SSE έχει μικρότερο χρόνο εκτέλεσης για αριθμό νημάτων ίσο με P = 4 , και ακόμα μικρότερο για P = 2. (Omega Time SSE – 0.03 ενώ Omega Time SSE+PTHREAD – 0.0289 (P=2), 0262(P=4)).