Μάθημα: Συστήματα Υπολογισμού Υψηλών Επιδόσεων (ΜΔΕ 646)

Ονοματεπώνυμο: Θεοδώρου Γεώργιος

ΑΕΜ: 0497

Report

Αρχικά έκανα ένα loop profiling του original κώδικα, για να εντοπίσω τα loop στα οποία καθυστερεί πιο πολύ η εκτέλεση του κώδικα μας. Τα πιο χρονοβόρα loop, ήταν εκείνα τα οποία είχαν πολλές επαναλήψεις. Αφού εντόπισα τα πιο χρονοβόρα loop, ξεκίνησα να ελέγχω πια από αυτά τα loop είναι παραλληλοποιήσιμα. Έπειτα σταδιακά άρχισα να παραλληλοποιώ καθένα από τα παραπάνω loop, ελέγχοντας κάθε φορά εάν χρειάζεται να θέσω κάποιες μεταβλητές private. Βέβαια όσο προχωρούσα σε λιγότερο χρονοβόρα loop, έβλεπα μείωση της επίδοσης, λόγο της επιβάρυνσης της δημιουργίας πολλαπλών thread.

Πρακτικά έχω παραλληλοποιήσει 9 for loop, τα οποία θεώρησα ότι δίνουν την βέλτιστη επίδοση στον κώδικά μας. Φυσικά για να καταλήξω στα συγκεκριμένα loop έκανα αρκετές δοκιμές και τροποποιήσεις, οι οποίες κατέληγαν σε χειρότερες επιδόσεις. Επιπλέον για να κάνω παράλληλα κάποια συγκεκριμένα for loop έθεσα τα numthreads του κώδικα ίσα με τον αριθμό των thread που χρησιμοποιούμε κάθε φορά και τη μεταβλητή my_cpu_id ίση με το εκάστοτε thread που εκτελεί το loop.

Όσο αναφορά τον χρόνο εκτέλεσης του κώδικα, παρατήρησα μεγάλη βελτίωση στις περιπτώσεις με τα 8 και 14 thread. Το μηχάνημα στο οποίο κάναμε τις μετρήσεις έχει 2 επεξεργαστές με 14 πυρήνες ο καθένας και είναι και hyperthreaded, επομένως αναγνώριζε ουσιαστικά 52 πυρήνες. Πιστεύω ότι είχα καλύτερη επίδοση όταν χρησιμοποιούσα 8 και 14 thread, διότι σε εκείνες τις περιπτώσεις είχα το λιγότερο overhead από το OpenMP σε σχέση με την ταχύτητα που κέρδιζα από την παραλληλοποίηση. Φυσικά όπως ήταν αναμενόμενο όταν έτρεξα τον παράλληλο κώδικα με ένα thread είδα δραματική αύξηση στον χρόνο εκτέλεσης του κώδικα, λόγο του overhead από το OpenMP. Επίσης όταν χρησιμοποίησα και τα 52 thread του μηχανήματος είδα σημαντική μείωση της επίδοσης του προγράμματος μας, και πιστεύω συνέβη διότι το overhead της δημιουργίας 52 thread ήταν υπερβολικά μεγάλο.

Τέλος μία μικρή παρατήρηση που έκανα κατά το compile, είναι ότι όταν χρησιμοποιούσα το –fast flag, ο χρόνος εκτέλεσης του προγράμματος αυξανόταν.

Ακολουθεί σχετικός πίνακας και σχεδιάγραμμα με την μέση τιμή και την τυπική απόκλιση από τον αρχικό κώδικα και τον τελικό. Οι χρόνοι είναι μετρημένοι για το μεγάλο input αρχείο.

	Original		OpenMP		
Threads	Average Time (s)	Divergence	Average Time (s)	Divergence	Improvement
1	9,0194	0,112182173	12,1064	0,073476799	-34%
2	-	-	7,4824	0,144203467	17%
4	-	-	4,4117	0,074366726	51%
8	-	-	3,1204	0,308306082	65%
14	-	-	2,9546	0,36535413	67%
28	-	-	3,3138	0,101283562	63%
56	-	-	5,0188	0,241245435	44%

Οι χρόνοι είναι μετρημένοι σε seconds.

Στο σχεδιάγραμμα που ακολουθεί, για να φαίνεται πιο καθαρά η διαφορά στην επίδοση, έχω θέσει τους χρόνους του αρχικού κώδικα ίδιους για όλα τα threads.