Συστήματα και Αλγόριθμοι Πολυμέσων

Ιωάννης Χαρ. Κατσαβουνίδης

Ομιλία #6: Αρχές Βελτιστοποίησης Λογισμικού Πολυμέσων

8 Νοεμβρίου 2005

Πανεπιστήμιο Θεσσαλίας
Τμήμα Μηχ. Η/Υ, Τηλεπ. & Δικτύων

Επανάληψη

- Θεωρία Πληροφορίας
  - Αριθμητική Κωδικοποίηση – CABAC
  - Θεωρία ρυθμού μετάδοσης – μείωσης ποιότητας
- Προ- και Μετά-επεξεργασία πολυμεσικών σημάτων
  - Αλλαγή συστήματος συντεταγμένων χρώματος
  - Διασπορά σφάλματος κβαντοποίησης (dithering)
  - Βίντεο εναλλασόμενων/διαδοχικών γραμμών

Επανάληψη (2)

- Προ- και Μετά-επεξεργασία πολυμεσικών σημάτων
  - Αλλαγή φωτεινότητας/χρώματος και υφής σήματος βίντεο
  - Φιλτράρισμα ασυνεχειών στα όρια τετραγώνων συμπίεσης (de-blocking, de-ringing)
  - Αλλαγή αριθμού καναλιών ήχου (mono-stereo, Dolby 5.1-stereo)
  - Αλλαγή ρυθμού αναπαραγωγής χωρίς αλλαγή συχνότητας ήχου (“time-stretching”)

Άσκηση #6

- Να μετατρέψετε το 1ο καρέ του src14 από το σύστημα συντεταγμένων YUV σε RGB, απεικονίζοντας τις τιμές RGB με 8 bits και μετά να το επαναφέρετε στο σύστημα YUV. Ποιο είναι το μέσο σφάλμα κατά τη μετατροπή ?
- Να επαναλάβετε την μετατροπή απεικονίζοντας τις τιμές RGB με 4 bits. Στη συνέχεια να εισάγετε διατεταγμένη διασπορά σφάλματος D_4.
- Παραδώστε τα αρχεία εξόδου (.YUV)

Βασικές τεχνικές (αρχές) επεξεργασίας πολυμέσων

- *Δειγματοληψία*
- *Κβαντοποίηση*
- *Μετασχηματισμοί*
- *Θεωρία Πληροφορίας*
- *Προ- και Μετά-επεξεργασία πολυμεσικών σημάτων*

Πειραματική μέτρηση απόδοσης και βελτιστοποίηση λογισμικού

- Οι αλγόριθμοι πολυμέσων είναι πολύπλοκοι υπολογιστικά
- Το βίντεο αποτελεί μια από τις πιο απαιτητικές εφαρμογές λογισμικού
  - Υψηλοί ρυθμοί πρόσβασης μνήμης, δικτύου και αποθηκευτικών συσκευών
  - Μεγάλες ποσότητες υπολογισμών ανά δευτερόλεπτο
  - Υψηλές απαιτήσεις πιστότητας εικόνας και ήχου
- Μεγάλη απήχηση στο ευρύ κοινό

Υπολογισμός πολυπλοκότητας

- Παράδειγμα: αποκωδικοποίηση βίντεο MPEG2, διαστάσεων W(width) x H(height) x f(frames/sec) x B (bitrate –bps)
  - Το πρώτο στάδιο της αποκωδικοποίησης Huffman απαιτεί 1 getbits(1) για κάθε bit. Η συνάρτηση getbits(1) χρειάζεται 1 AND + 1 SHIFT => B*2 ops/sec
  - Το δεύτερο στάδιο της αντίστροφης κβαντοποίησης απαιτεί 1 πολλαπλασιασμό για κάθε συντελεστή DCT, συνεπώς W*H*f*1.5 ops/sec

Υπολογισμός πολυπλοκότητας (2)

- Αποκωδικοποίηση βίντεο MPEG2, διαστάσεων W(width) x H(height) x f(frames/sec) x B (bitrate – bps) – συνέχεια
  - Το τρίτο στάδιο είναι το IDCT, το οποίο απαιτεί log2(8) + log2(8) = 6 ops/sec για κάθε pixel => W*H*1.5*f*6 = 9*W*H*f ops/sec
  - Το επόμενο στάδιο είναι ο υπολογισμός των pixels του πίνακα αναφοράς (παρεμβολή), το οποίο απαιτεί 0 (ακέραια pixels), 3 (παρεμβολή μόνο κατά τη μία κατεύθυνση) ή 6 (παρεμβολή κατά τις δύο κατευθύνσεις) πράξεις για κάθε pixel => 3 ops/sec για κάθε pixel => 4.5*W*H*f ops/sec
  - Το τελευταίο στάδιο είναι η πρόσθεση των pixels του πίνακα αναφοράς με τις τιμές εξόδου του IDCT, το οποίο απαιτεί 1 op/sec για κάθε pixel => W*H*1.5*f ops/sec

Υπολογισμός πολυπλοκότητας (3)

- Η κάθε πράξη (operation) χρειάζεται πολλαπλούς κύκλους μηχανής, καθότι υπάρχει η καθυστέρηση λόγω μεταφοράς δεδομένων από και προς τη μνήμη. Υποθέτοντας 4 κύκλους για κάθε πράξη, τότε έχουμε
- 8*B + 66*W*H*f cycles/sec
  - DVD (720x480x30x8.0): 748M cycles/sec

Υπολογισμός πολυπλοκότητας (4)

- Στην περίπτωση της τηλεόρασης υψηλής ευκρίνειας (HDTV), ο ίδιος υπολογισμός δίνει ~ 4.5GHz
- Στην πράξη, τα πράγματα είναι χειρότερα από τον προηγούμενο υπολογισμό, καθότι υπάρχει μεγάλο overhead
  - Μεταφορά δεδομένων από πίνακα σε πίνακα
  - Καθυστέρηση λόγω πρόσβασης αποθηκευτικών μέσων (σκληρός δίσκος, οπτικός δίσκος)
  - Μετα-επεξεργασία, όπως αλλαγή συστήματος συντεταγμένων χρώματος, επαναδειγματοληψία κλπ.

Βελτιστοποίηση λογισμικού

- Είναι λοιπόν απαραίτητη η βελτιστοποίηση του λογισμικού πολυμέσων για να καταφέρουμε να έχουμε τη μέγιστη δυνατή βάση εγκατάστασης της εφαρμογής μας
  - WinDVD: αναπαραγωγή DVD χρησιμοποιώντας περίπου 200MHz Pentium-II

Βελτιστοποίηση λογισμικού (2)

- Τεχνικές βελτιστοποίησης
  - Αλγοριθμικά, π.χ. η χρήση FDCT αντί για τον ορισμό του (πολλαπλασιασμός πίνακα με διάνυσμα)
  - Μείωση στο ελάχιστο δυνατό της χρήσης μνήμης και μεγιστοποίηση της επανα-χρησιμοποίησης ενδιάμεσων αποτελεσμάτων σε καταχωρητές
  - Χρήση εξειδικευμένων εντολών, όπως η οικογένεια των εντολών Μονή-Εντολή-Σε-Πολλαπλά-Δεδομένα (SIMD – MMX/SSE/SSE2)
  - Χρήση επιπρόσθετου υλικού, όπως η κάρτα γραφικών, για την επιτάχυνση ορισμένων κομματιών του αλγόριθμου

Βελτιστοποίηση λογισμικού (3)

- Απαραίτητο βήμα για τη βελτιστοποίηση λογισμικού αποτελεί η συλλογή πειραματικών δεδομένων που να αντικατοπτρίζουν την πολυπλοκότητα της εφαρμογής στο σύστημα το οποίο επιθυμούμε να τρέξουμε την εφαρμογή μας.
- Το βήμα αυτό λέγεται “profiling”

Βελτιστοποίηση λογισμικού (4)

- Τα εργαλεία που μας επιτρέπουν να κάνουμε profiling είναι διάφορα και εξαρτώνται τόσο από το σύστημα που χρησιμοποιούμε (PC/embedded device/ASIC) όσο και από το λειτουργικό σύστημα (Windows/Linux/u-Kernel)
- Το πιο δημοφιλές εργαλείο για profiling σε περιβάλλον PC-Windows-Intel λέγεται Vtune και διατίθεται μέσω της εταιρίας Intel

Intel VTune

- Η πιο ανανεωμένη έκδοσή του (Νοεμ. 2005) είναι η 7.2
- Επιτρέπει τη συλλογή πειραματικών στοιχείων για την κατανόηση της πολυπλοκότητας οποιαδήποτε εφαρμογής
- Τρέχει σε λειτουργικά συστήματα Windows και Linux
- Τρέχει σε όλους της μικρο-επεξεργαστές της Intel, δηλ. Pentium, Xscale, Itanium

Intel VTune (2)

- Έχει δύο βασικούς τρόπους συλλογής δεδομένων
  - Δειγματοληπτικά (Sampling)
  - Εξουθενωτικά (Exhaustive – call graph)

Intel VTune (3)

- Δειγματοληπτική συλλογή δεδομένων μας επιτρέπει να γνωρίζουμε σε ποια κομμάτια της εφαρμογής μας «ξοδεύουμε» το μεγαλύτερο ποσοστό του χρόνου εκτέλεσης
- Εξουθενωτική συλλογή δεδομένων μας επιτρέπει να γνωρίζουμε την ακριβή δομή και σχέση μεταξύ κλητή (caller) και καλούμενης συνάρτησης (callee functions) μαζί με ακριβή αριθμό κλήσεων.

Intel VTune (4)

- Η δειγματοληπτική συλλογή βασίζεται στη στατιστική για την ακρίβεια των αποτελεσμάτων, χωρίς να αλλοιώνει την εκτέλεση του προγράμματος.
- Η εξουθενωτική συλλογή είναι εξ'ορισμού ακριβής, όμως αλλοιώνει κατά μεγάλο βαθμό την εκτέλεση της εφαρμογής μας εισάγοντας «ξένο» κομμάτι κώδικα για τον αποκλειστικό σκοπό τη συλλογή της απαραίτητης πληροφορίας.

Εξουθενωτική συλλογή δεδομένων εφαρμογών

- Συνήθως εφαρμόζεται στην αρχή της ανάπτυξης μιας εφαρμογής με σκοπό
  - Την κατανόηση του πηγαίου κώδικα
  - Την κατανόηση της συχνότητας εκτέλεσης των διαφόρων συναρτήσεων του προγράμματος
  - Την κατανόηση της σχέσης καλούντος (caller) και κληθείσας (callee) συνάρτησης
  - Την απόκτηση μιας πρώτης εικόνας της πολυπλοκότητας των διαφόρων συναρτήσεων

Δειγματοληπτική συλλογή δεδομένων εφαρμογών

- Υπάρχουν 2 τρόποι δειγματοληψίας
  - Με τη βοήθεια του ρολογιού του συστήματος
  - Με τη βοήθεια εσωτερικών καταχωρητών/μετρητών του μικρο-επεξεργαστή
- Και οι δύο μέθοδοι χρησιμοποιούν interrupts
  - Η συχνότητα του interrupt καθορίζεται από το χρήστη, όμως συνήθως είναι 1KHz (=1 δείγμα κάθε 1 msec)

Δειγματοληπτική συλλογή δεδομένων εφαρμογών (2)

- Τα interrupts σταματούν την εκτέλεση του προγράμματός μας και αποθηκεύουν την τιμή του μετρητή εντολών προγράμματος (PC)
- Αφού ολοκληρωθεί η εφαρμογή, ή εφόσον έχουμε συλλέξει αρκετά δείγματα, οι τιμές του PC που έχουν αποθηκευτεί αντιστοιχίζονται στις διάφορες συναρτήσεις του προγράμματος με τη βοήθεια κάποιας βάσης δεδομένων (.pdb)
- Έτσι λοιπόν έχουμε την κατανομή του χρόνου εκτέλεσης στις διάφορες συναρτήσεις του προγράμματος

Δειγματοληπτική συλλογή δεδομένων εφαρμογών (3)

- Επιπλέον, μπορούμε να αποκτήσουμε και την κατανομή του χρόνου εκτέλεσης στις διάφορες εκτελέσιμες εντολές της κάθε συνάρτησης
- Κατ' αυτό τον τρόπο, μπορούμε να επικεντρώσουμε την προσοχή μας στα κομμάτια του κώδικα που είναι υπεύθυνα για το μεγαλύτερο ποσοστό του χρόνου εκτέλεσης.
- Ξανα-γράφοντας τα κομμάτια αυτά μπορούμε να πετύχουμε τη μέγιστη βελτίωση του χρόνου εκτέλεσης της εφαρμογής μας

Δειγματοληπτική συλλογή δεδομένων εφαρμογών (4)

- Η διαδικασία αυτή επαναλαμβάνεται κάθε φορά που έχουμε επιφέρει αλλαγές στον αλγόριθμο ή στην υλοποίησή του
  - Για να επιβεβαιώσουμε ότι οι αλλαγές είναι προς την σωστή κατεύθυνση
  - Για να επιλέξουμε από πιθανώς περισσότερων από μία εναλλακτικών λύσεων

Δειγματοληπτική συλλογή δεδομένων εφαρμογών (5)

- Η χρήση του ρολογιού του συστήματος είναι η πιο αντικειμενική για την αξιολόγηση της συνολικής πολυπλοκότητας (=χρόνος εκτέλεσης) των διάφορων συναρτήσεων του προγράμματος
- Η χρήση των εσωτερικών καταχωρητών/μετρητών του μικρο-επεξεργαστή επιτρέπει να μελετήσουμε με λεπτομέρεια το λόγο της αυξημένης πολυπλοκότητας, όπως
  - Μεγάλος αριθμός εντολών
  - Χρήση εντολών αυξημένης πολυπλοκότητας
  - Υπερβολική ή μη-βέλτιστη χρήση μνήμης

Δειγματοληπτική συλλογή με χρήση εσωτερικών μετρητών

- Οι περισσότεροι μικρο-επεξεργαστές γενικής χρήσης έχουν μια σειρά εσωτερικών καταχωρητών οι οποίοι δεν είναι διαθέσιμοι για γενικές εφαρμογές, παρά μόνο για την παρακολούθηση ορισμένων εσωτερικών σημάτων, όπως
  - Ο αριθμός εντολών που εκτελέστηκαν
  - Ο αριθμός των εσωτερικών κύκλων ρολογιού
  - Ο αριθμός των προσπελάσεων εξωτερικής μνήμης μέσω της γρήγορης μνήμης cache

Δειγματοληπτική συλλογή με χρήση εσωτερικών μετρητών (2)

- Οι εσωτερικοί αυτοί καταχωρητές μπορούν να προγραμματιστούν ως μετρητές αυτών των εσωτερικών σημάτων
- Εν συνεχεία, μπορούμε να προγραμματίσουμε ένα όριο για κάθε ένα από αυτούς τους μετρητές – όταν η τιμή του μετρητή φτάσει το όριο, δημιουργείται μια διακοπή (interrupt), η οποία σταματά την κανονική εκτέλεση του προγράμματος και οδηγείται στην εκτέλεση μιας ρουτίνας εξυπηρέτησης διακοπής (ISR)
- Στην περίπτωση του VTune, αυτή η ρουτίνα απλά καταχωρεί την τιμή του μετρητή προγράμματος (PC)

Δειγματοληπτική συλλογή με χρήση εσωτερικών μετρητών (3)

- Οι πιο χρήσιμοι μετρητές είναι αυτοί των εσωτερικών κύκλων ρολογιού και του αριθμού των εκτελέσιμων εντολών
- Ο πρωταρχικός μας σκοπός είναι να μειώσουμε στο ελάχιστο δυνατό τον αριθμό των κύκλων ρολογιού
- Συνήθως, μειώνοντας τον αριθμό τον εντολών, μειώνεται και ο αριθμός των κύκλων ρολογιού
  - Εκτός αν χρησιμοποιούμε πολύπλοκες εντολές
  - Ή έχουμε μεγάλη καθυστέρηση λόγω πρόσβασης μνήμης
- Ένας πολύ σημαντικός δείκτης «ποιότητας» του κώδικα είναι ο λόγος εσωτερικών κύκλων ρολογιού του μικροεπεξεργαστή προς τον αριθμό των εκτελέσιμων εντολών (CPI)

Δειγματοληπτική συλλογή με χρήση εσωτερικών μετρητών (4)

- Πολύ σημαντικός μετρητής είναι και αυτός του αριθμού των αποτυχιών εύρεσης δεδομένων στην γρήγορη μνήμη (cache), το οποίο συνεπάγεται πρόσβαση της αργής εξωτερικής μνήμης του συστήματος (RAM). Αυτός ο μετρητής ονομάζεται L2 Cache Read Misses.
- Η ελαχιστοποίηση των προσπελάσεων αργής μνήμης είναι το «μυστικό» της βελτιστοποίησης λογισμικού πολυμέσων

Δειγματοληπτική συλλογή με χρήση εσωτερικών μετρητών (5)

- Για να υπολογιστεί η σωστή τιμή ορίου του μετρητή κατά τρόπο ώστε η συχνότητα δειγματοληψίας να είναι η ζητούμενη (συνήθως 1KHz), το VTune εκτελεί το πρόγραμμα 2 φορές
  - Την πρώτη φορά απλά συλλέγει το συνολικό αριθμό των συμβάντων που επιθυμούμε να συλλέξουμε (π.χ. αριθμός εκτελέσιμων εντολών). Αυτή η πρώτη εκτέλεση λέγεται calibration run. Διαιρώντας το συνολικό αριθμό συμβάντων με τον συνολικό χρόνο εκτέλεσης υπολογίζει ποιο πρέπει να είναι το όριο για τη δημιουργία διακοπής (interrupt)

Δειγματοληπτική συλλογή με χρήση εσωτερικών μετρητών (6)

- Για τη σωστή ερμηνεία των αποτελεσμάτων του VTune, πρέπει να δημιουργήσουμε όλα τα σύμβολα κατά τη διαδικασία μεταγλώττισης του πηγαίου κώδικα.
  - Αυτό γίνεται με την επιλογή του “C/C++ -> General -> Debug Information Format/Program database (/Zi)” στον περιβάλλον Microsoft .NET 2003
  - Επίσης πρέπει να ενεργοποιήσουμε την αντίστοιχη επιλογή στο “Linker -> Debugging -> Generate Debug Ingo/Yes (/DEBUG)” καθώς και να εισάγουμε “/fixed:NO” στο “Linker -> Command Line -> Additional Options”

Άσκηση

- Να μεταφέρετε το πρόγραμμα αναφοράς αποκωδικοποίησης βίντεο MPEG2 (“MPEG2DEC”) σε περιβάλλον Windows/.NET 2003
- Να μετατρέψετε την έξοδο του προγράμματος κατά τρόπο που να δημιουργεί ένα ενιαίο αρχείο εξόδου .YUV
- Να επιτρέψετε την εκτέλεση του προγράμματος τόσο με δημιουργία αρχείου εξόδου (π.χ. Με τη χορήγηση των παραμέτρων “-b Dantes_2_00M_cut.m2v -f -g -o0 Dantes_2_00M_cut.yuv”), όσο και χωρίς τη δημιουργία αρχείου εξόδου
- Να αναφέρετε το συνολικό αριθμό καρέ που αποκωδικοποιήθηκαν, το χρόνο εκτέλεσης και την ταχύτητα αποκωδικοποίησης (frames per sec, fps)

Άσκηση (συνέχεια)

- Να δοκιμάσετε τις διάφορες επιλογές βελτιστοποίησης κατά τη διαδικασία μεταγλωττισμού που σας δίνει το περιβάλλον .NET 2003
- Να αντικαταστήσετε το μεταγλωττιστή Microsoft VC7 με τον μεταγλωττιστή Intel Compiler 9.0 (IC9) και να δοκιμάσετε τις διάφορες επιλογές βελτιστοποίησης
- Να αναφέρετε την μεγαλύτερη ταχύτητα εκτέλεσης που επιτυγχάνετε από το VC7 και το (IC9)
- Να συλλέξετε δεδομένα με το πρόγραμμα Vtune για αυτή την εφαρμογή