Προγραμματισμός GPU σε περιβάλλον OpenCL

Πολλαπλασιασμός πινάκων και ταύτιση αλφαριθμητικών

Πυργιώτης Θεμιστοκλής
t.pirgiot@gmail.com

ΠΜΣ Τμήματος Εφαρμοσμένης Πληροφορικής
Συστήματα Υπολογιστών
Πανεπιστημίο Μακεδονίας

.

Επιβλέπων καθηγητής: Μαργαρίτης Κωνσταντίνος

Μάρτιος, 2012

Περιεχόμενα

Τι είναι το OpenCL;

Ορισμός

Το OpenCL (Open Computing Language) είναι ένα ανοιχτό πρότυπο για την υλοποίηση εφαρμογών γενικού σκοπού, οι οποίες εκτελούνται σε ετερογενείς πλατφόρμες αποτελούμενες από CPU, GPU και άλλους επεξεργαστές, δίνοντας έτσι την δυνατότητα στους προγραμματιστές να εκμεταλλευτούν στο έπακρο όλη την υπολογιστική ισχύ ενός συστήματος.

Τι είναι το OpenCL;

Αποτελείται από δύο βασικά στοιχεία:

- Το **API** που χρησιμοποιείται για τον συντονισμό των παράλληλων εφαρμογών σε ετερογενείς επεξεργαστές.
- Μια γλώσσα προγραμματισμού ανεξαρτήτου πλατφόρμας , την **OpenCL C** (βασιζόμενη στην C99).

Τι είναι το OpenCL;

Πλεονέκτηματα:

- Ανάπτυξη κώδικα ανεξαρτήτου πλατφόρμας χωρίς την ανάγκη χρησιμοποίησης γλώσσας συγκεκριμένου κατασκευαστή ή αντιστοίχισης με άλλες βιβλιοθήκες (OpenGL, DirectX).
- Μπορεί να δια λειτουργεί και να συνεργάζεται με το OpenGL, το OpenGL ES και άλλα γραφικά API.

Βασικοί όροι του OpenCL

- Platform *(host, devices)*
- Device *(GPU, multi-core CPU)*
- Kernel (*__kernel)*
- Context (*Διαχειρίζεται αντικείμενα του OpenCL*)
- Command Queue
- Memory Objects (*buffer*, *image*)
- Program

Αρχιτεκτονική του OpenCL

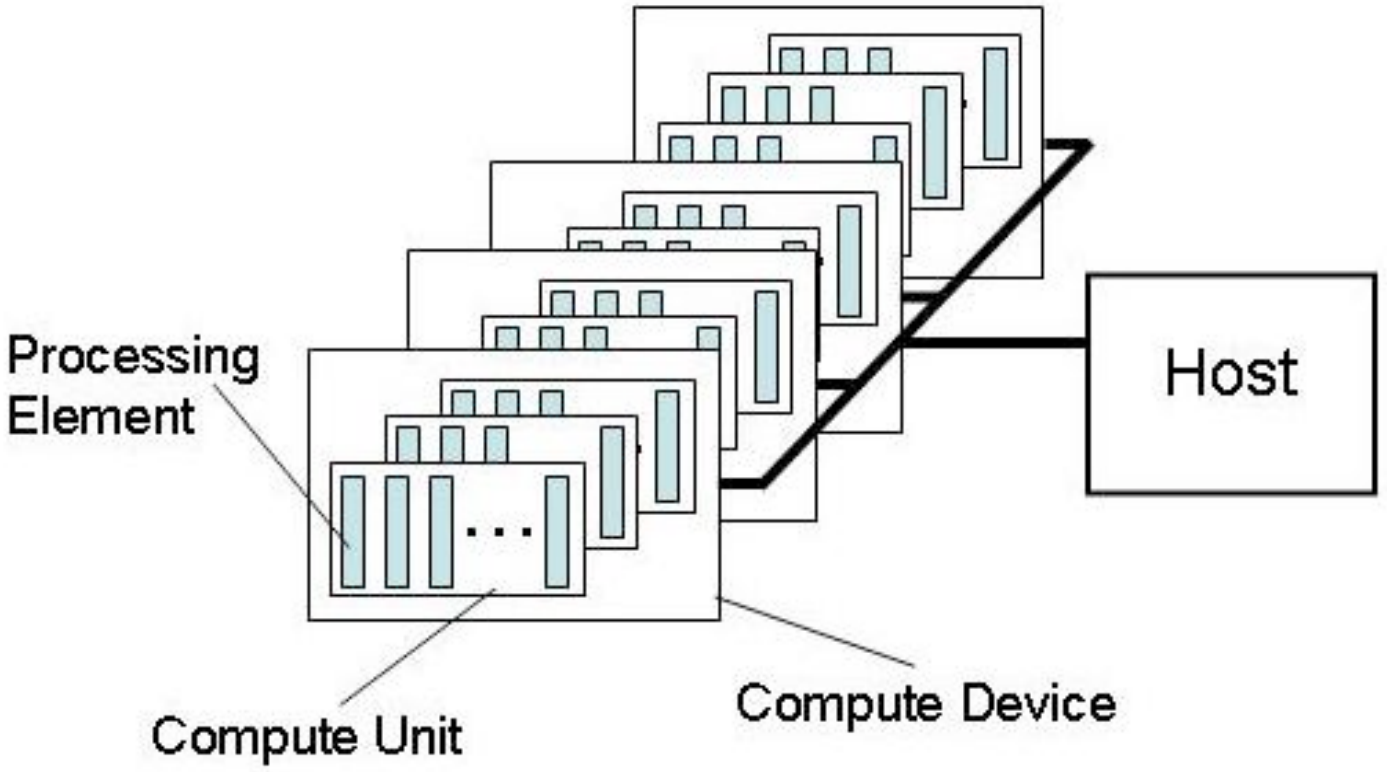
Παρουσιάζεται μέσα από μία ιεραρχία μοντέλων.

- Platform Model
- Execution Model
- Memory Model
- Programming Model

Αρχιτεκτονική του OpenCL

Platform Model

Το μοντέλο πλατφόρμας παρέχει μία υψηλού επιπέδου αφαίρεση για την OpenCL αρχιτεκτονική.



Αρχιτεκτονική του OpenCL

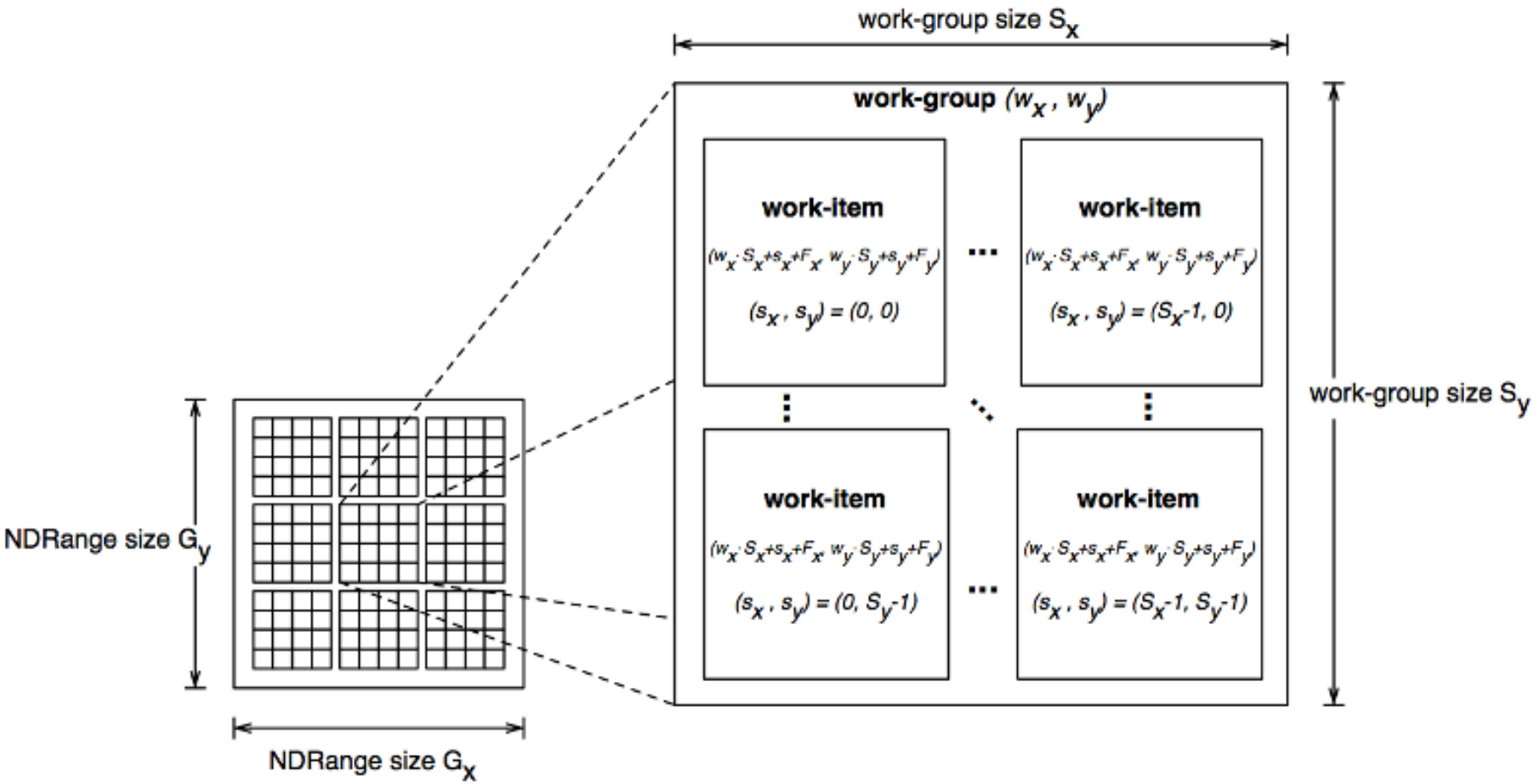
Execution Model

Η βασική αρμοδιότητα που έχει το μοντέλο εκτέλεσης, είναι ο καθορισμός εκτέλεσης των kernel στις συσκευές.

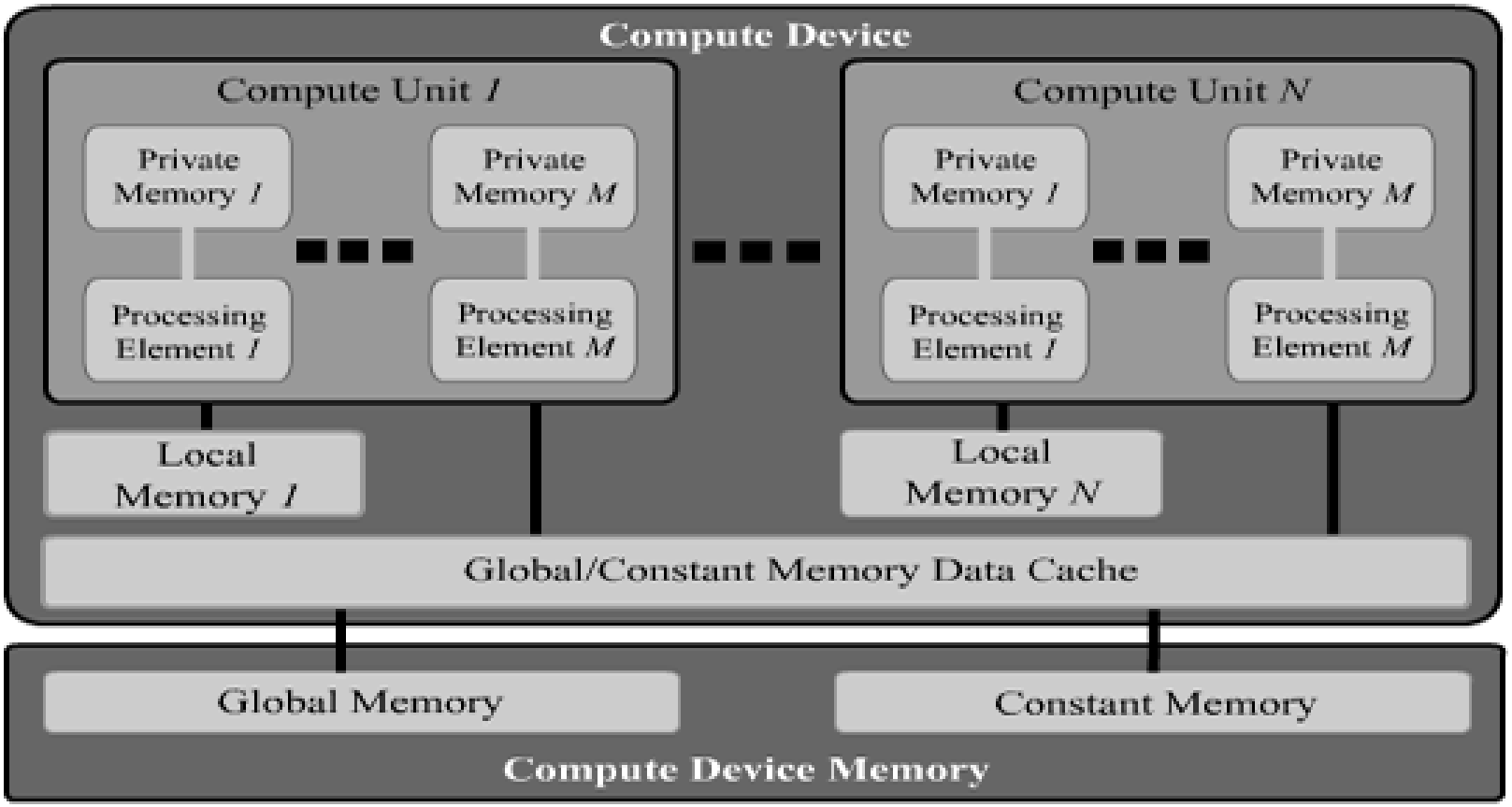
- work-item
- index space (NDRange), N = 1, 2, 3
- work-group
- global ID
- work-group ID
- local ID

Αρχιτεκτονική του OpenCL

Execution Model



Αρχιτεκτονική του OpenCL
Memory Model



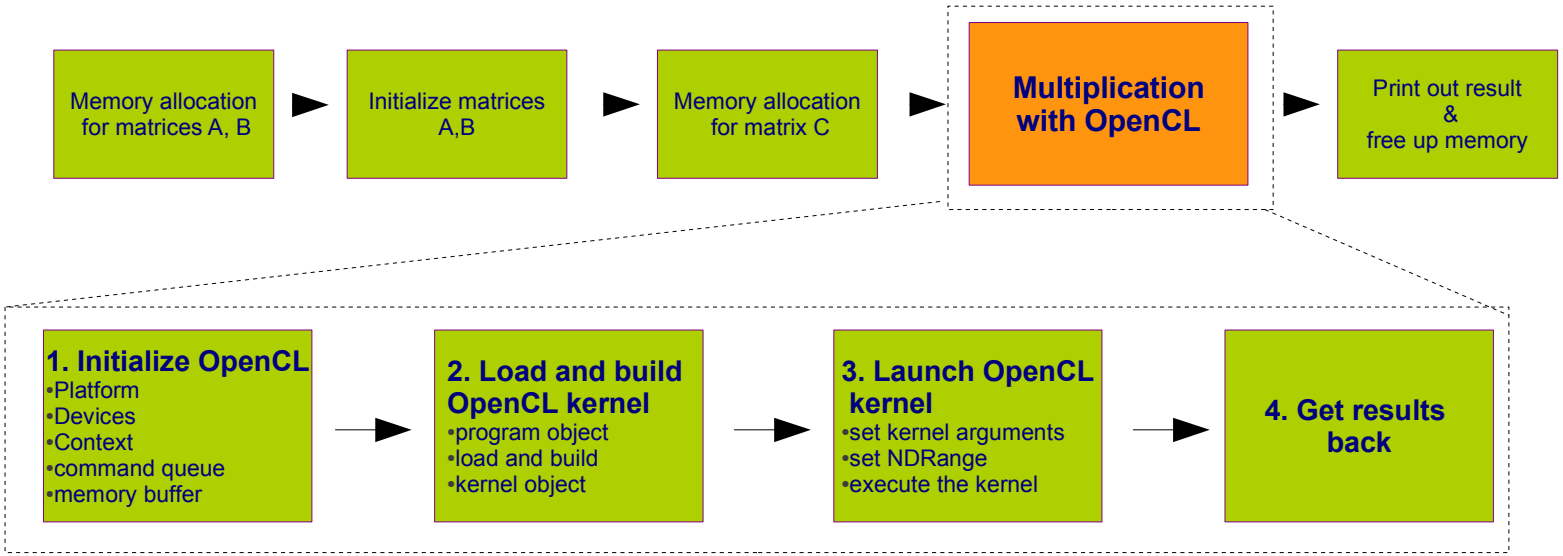
Αρχιτεκτονική του OpenCL
Programming Model

- **Data parallel**
  - Άμεσα, όπου ο προγραμματιστής καθορίζει τον συνολικό αριθμό work-item που θα εκτελεστούν παράλληλα και το πώς αυτά θα καταταχθούν στις διάφορες ομάδες εργασιών (work-group).
  - Εμμεσα, όπου ο προγραμματιστής καθορίζει μόνο τον συνολικό αριθμό των work-item που θα εκτελεστούν παράλληλα και το OpenCL από μόνο του αναλαμβάνει την κατάταξη αυτών στα διάφορα work-group.
- **Task parallel**

Περιεχόμενα

Πολλαπλασιασμός πινάκων

Πρόγραμμα εκτέλεσης στο host (C)



- `cl_mem d_A = clCreateBuffer( context , CL_MEM_READ_WRITE|CL_MEM_COPY_HOST_PTR, mem_size_A , h_A , &errcode );`
- ```
  size_t localWorkSize [2] , globalWorkSize [2];
  localWorkSize [0] = 16; localWorkSize [1] = 16;
  globalWorkSize [0] = 1024; globalWorkSize [1] = 1024;
  clEnqueueNDRangeKernel ( CQ , clKernel , 2 , NULL , globalWorkSize , localWorkSize , 0 , NULL , NULL ) ;
  ```
- `clEnqueueReadBuffer ( CQ , d_C , CL_TRUE , 0 , mem_size_C , h_C , 0 , NULL , NULL ) ;`
  ```

Πολλαπλασιασμός πινάκων

Πρόγραμμα εκτέλεσης στη συσκευή (OpenCL C, *.cl)

```
__kernel void matrixMul( __global float* C, __global float* A, __global float* B,
                         int wA, int wB){

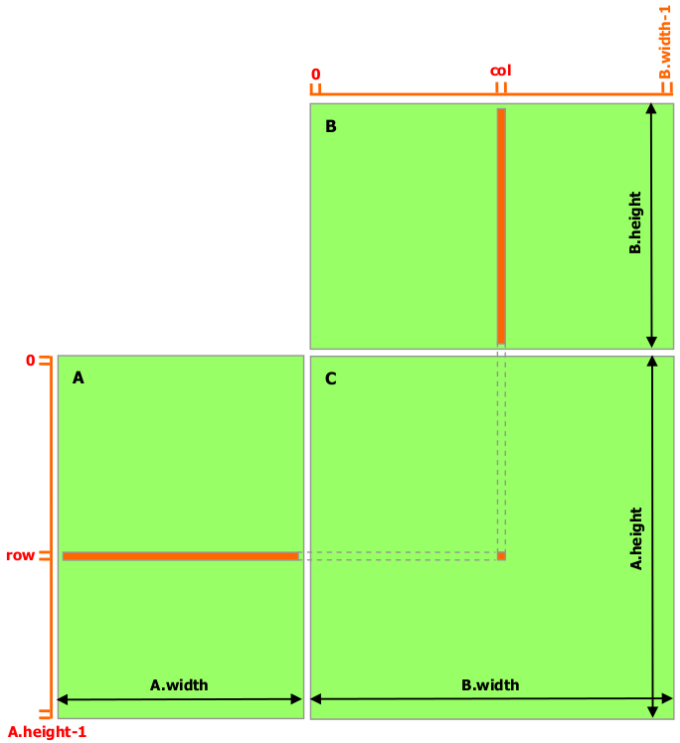
   // 2D Thread ID
   int tx = get_global_id(0);
   int ty = get_global_id(1);

   // value stores the element that is computed by the thread
   float value = 0;
   for (int k = 0; k < wA; ++k) {
      float elementA = A[ty * wA + k];
      float elementB = B[k * wB + tx];
      value += elementA * elementB;
   }

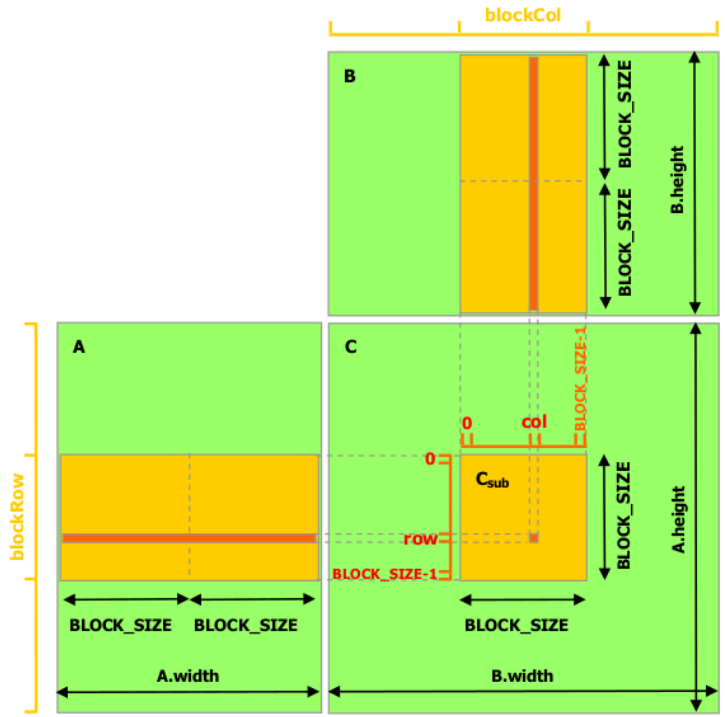
   // Write the matrix to device memory each thread writes one element
   C[ty * wA + tx] = value;
}
```

Πολλαπλασιασμός πινάκων

Χρήση global μνήμης

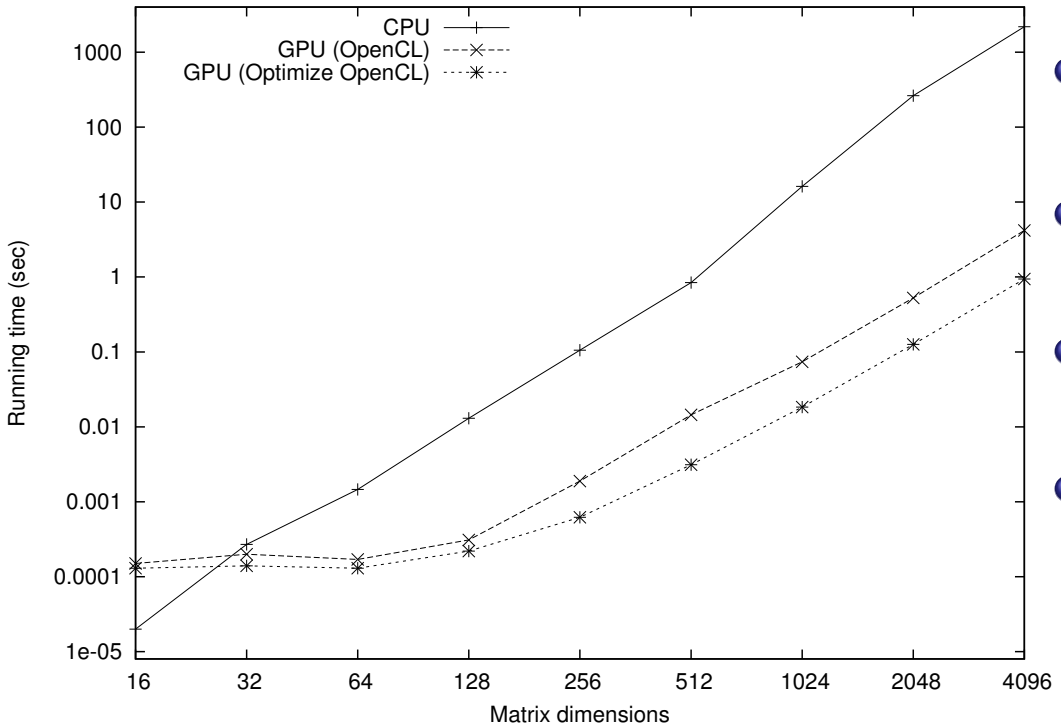


Χρήση local μνήμης



Πολλαπλασιασμός πινάκων

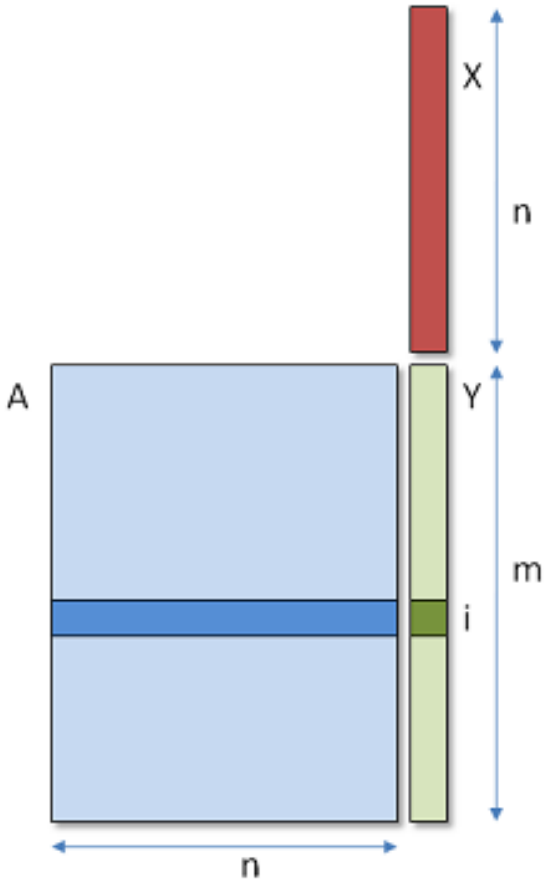
Αποτελέσματα πειραμάτων

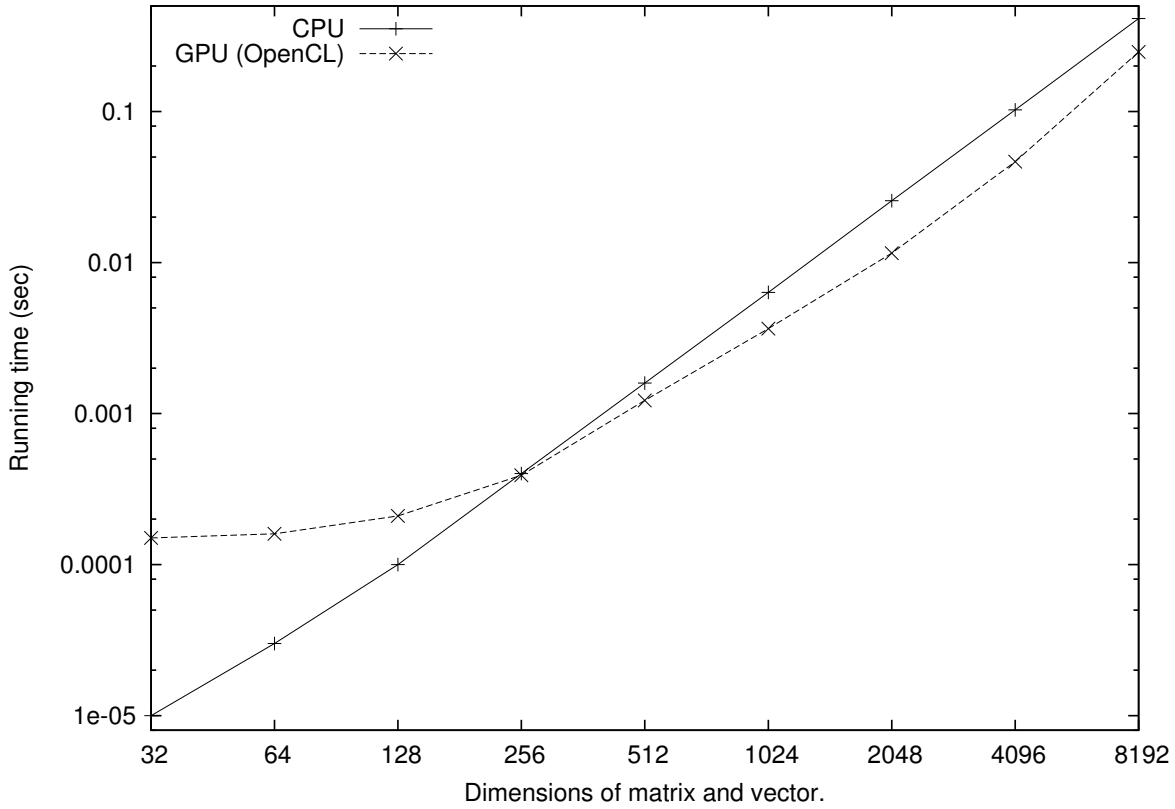


- 512, επιτάχυνση 52X (OpenCL, CPU)
- 1024, επιτάχυνση 219X (OpenCL, CPU)
- 2048, επιτάχυνση 4X (OpenCL opt, OpenCL)
- 1024, επιτάχυνση 881X (OpenCL opt, CPU)

Πίνακα με διάνυσμα

Αποτελέσματα πειραμάτος





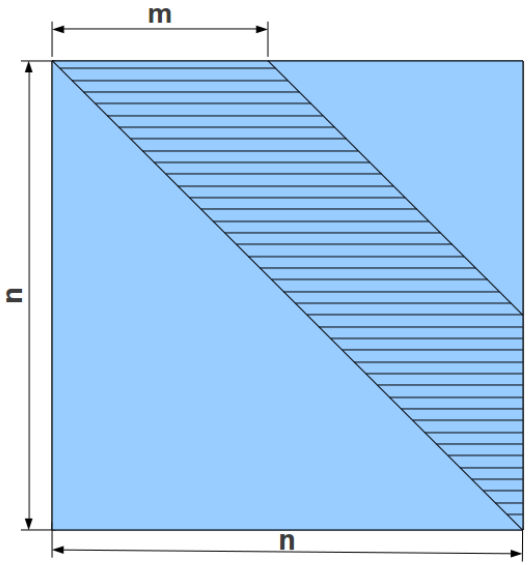
Πίνακα με διάνυσμα

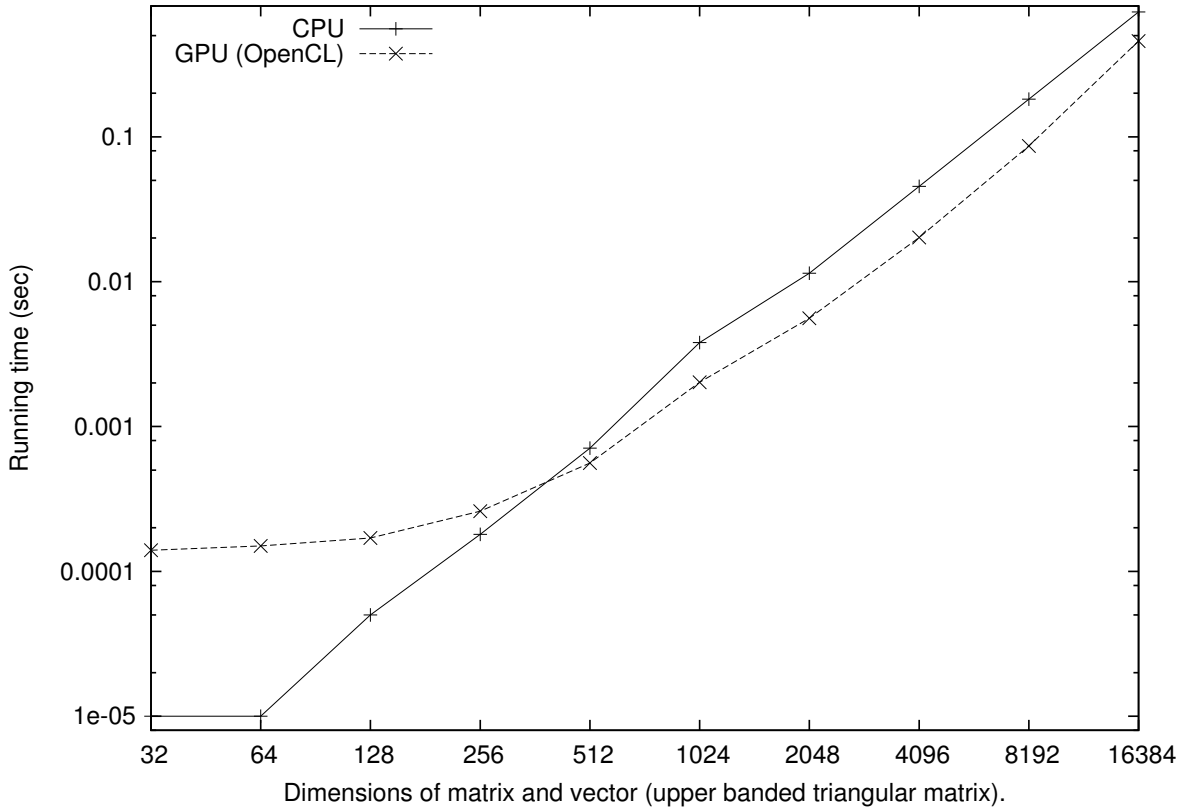
Αποτελέσματα πειραμάτος

- 1024, επιτάχυνση 1.7X (OpenCL, CPU)
- 2048, επιτάχυνση 2.2X (OpenCL, CPU)
- Στη περίπτωση πολλαπλασιασμού πινάκων ο αριθμός των πράξεων είναι ανάλογος του n^3, ενώ στο τρέχον παράδειγμα είναι ανάλογος του n^2.

Άνω τριγωνικού ζωνικού πίνακα με διάνυσμα

Αποτελέσματα πειραμάτος

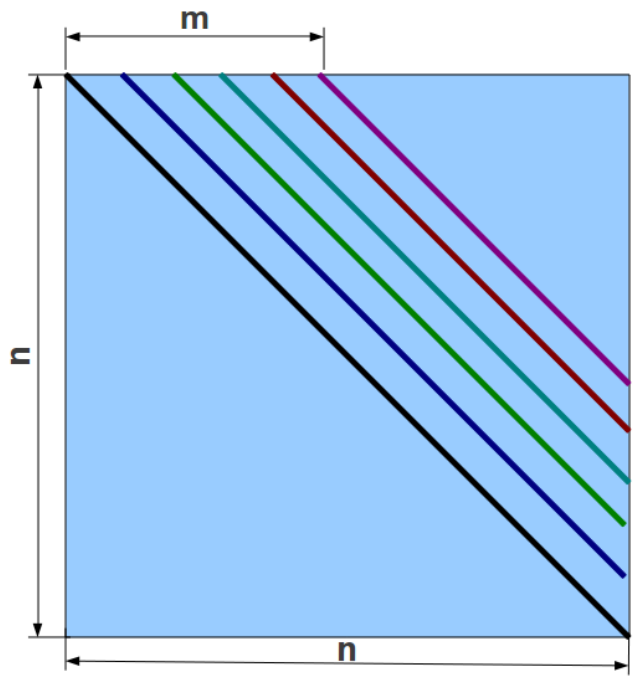


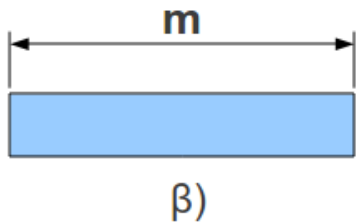


Το m, δηλαδή οι διαγώνιοι των πινάκων είναι $\frac{n}{2}$.

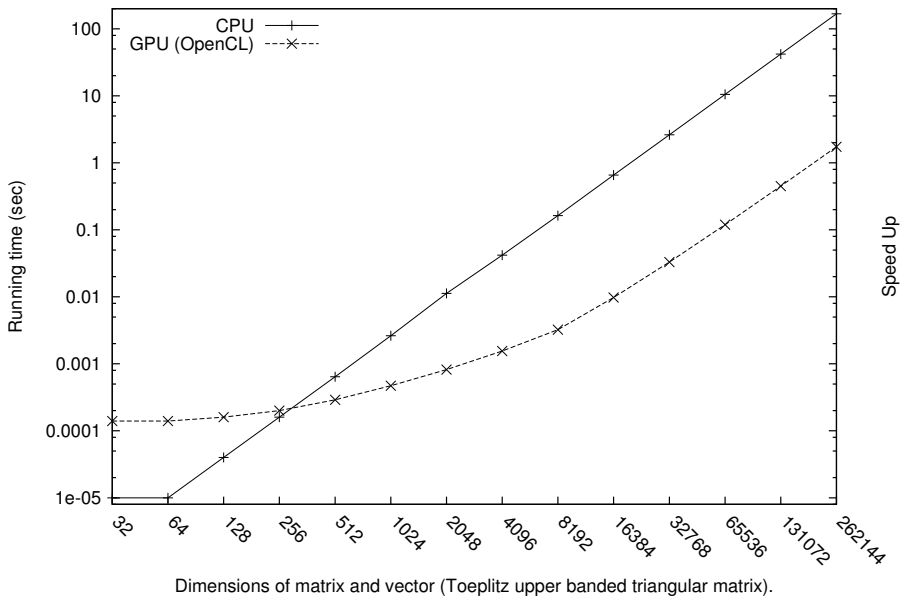
Άνω τριγωνικού ζωνικού πίνακα Toeplitz με διάνυσμα

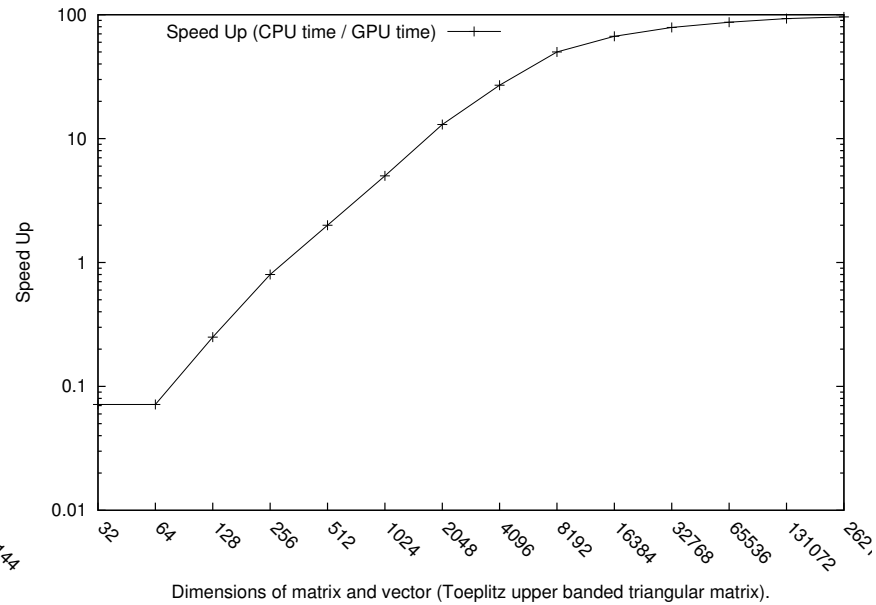
Ένας πίνακας Toeplitz έχει σε κάθε διαγώνιο σταθερά δεδομένα.





Άνω τριγωνικού ζωνικού πίνακα Toeplitz με διάνυσμα



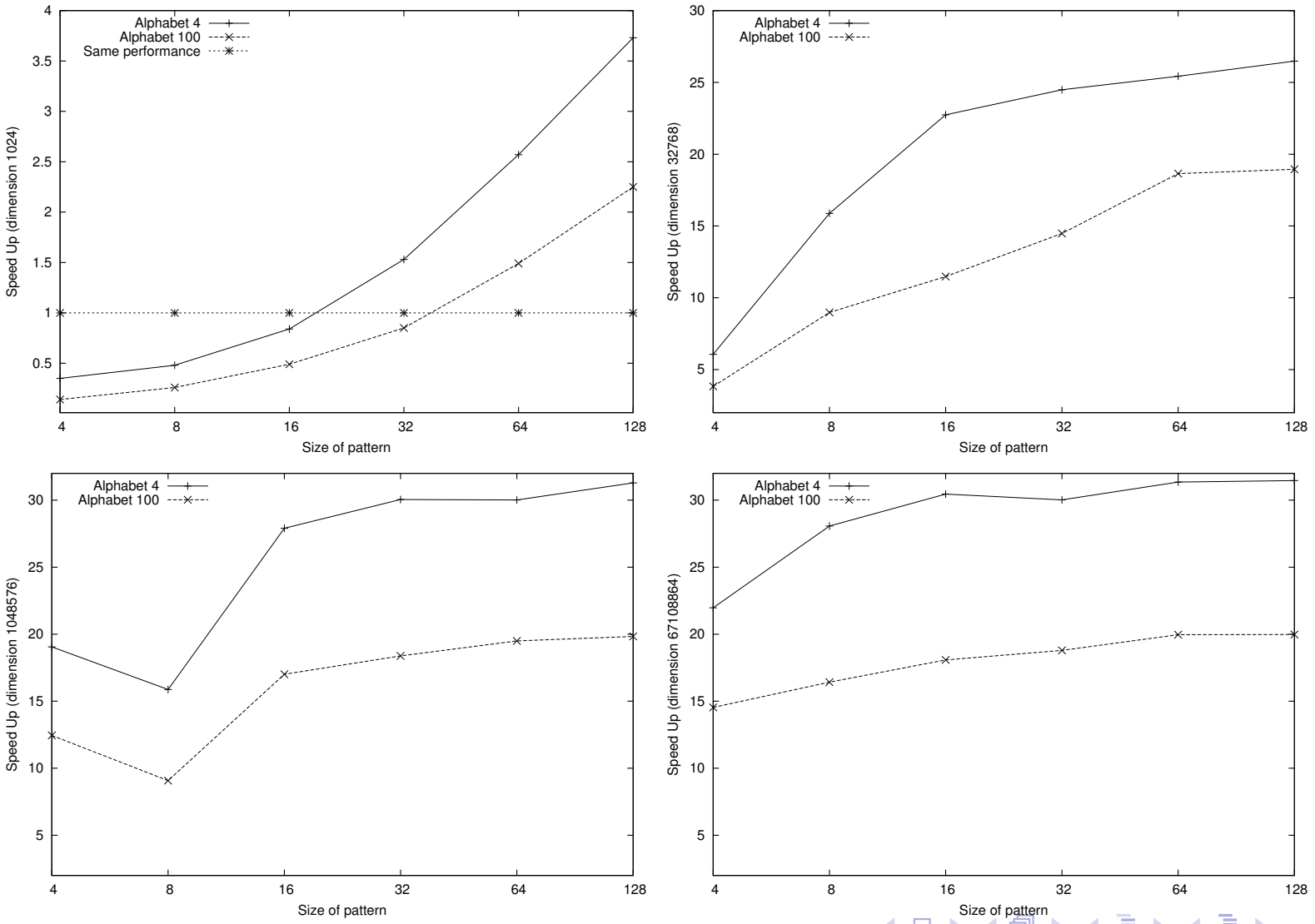


- Λόγω της σύμπτυξης του πίνακα A, τα πειράματα μπορούν να εκτελεστούν για πολύ μεγαλύτερες διαστάσεις.
- Η επιτάχυνση πλησιάζει στην τιμή 100.
- Λιγότερες προσπελάσεις στη μνήμη για τον πίνακα A, m στοιχεία και όχι $m \times n$.

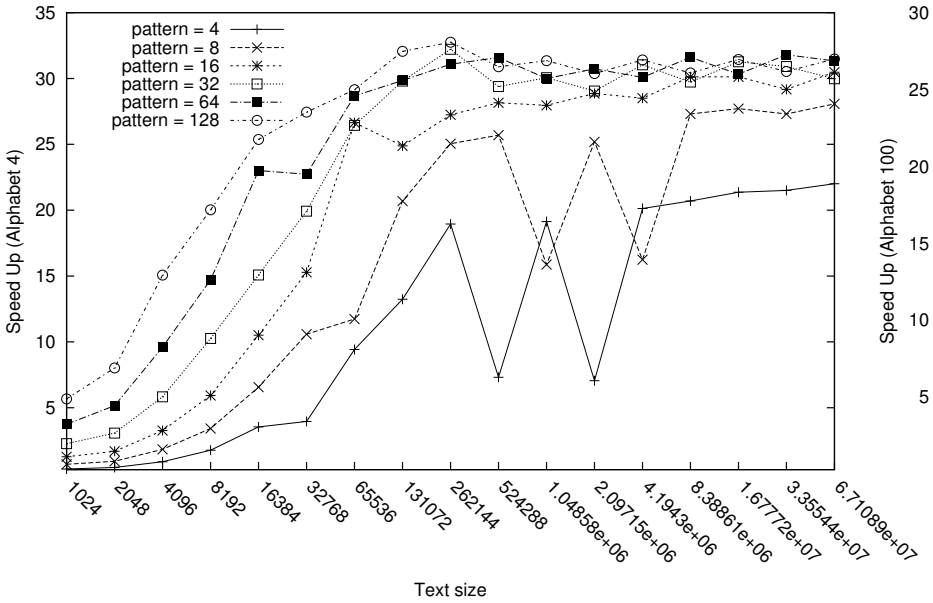
Προσεγγιστική Αναζήτηση προτύπου a σε κείμενο x

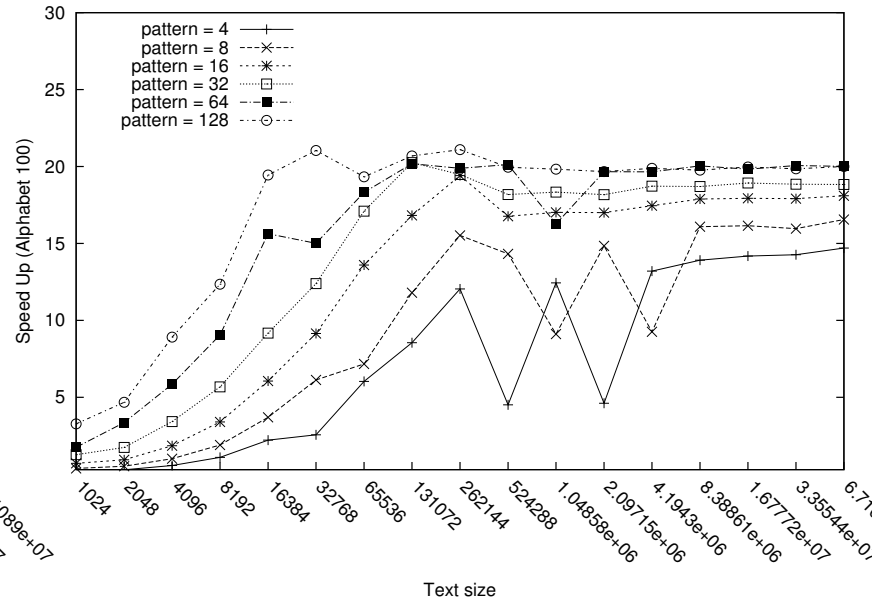
- Στο πίνακα a(διάνυσμα a) κρατείται το πρότυπο (μεγέθους 4, 8, 16, 32, 64, 128).
- Στο πολλαπλασιαζόμενο διάνυσμα x κρατείται το κείμενο πάνω στο οποίο θα γίνει η αναζήτηση.
- Η πράξη του πολλαπλασιασμού γίνεται έλεγχος ισότητας.
- Τύπος δεδομένων από float σε char.
- Απαιτούνται $n-m$ νήματα.
- Χρησιμοποιήθηκαν δύο αλφάβητα μεγέθους 4 και 100.

Προσεγγιστική Αναζήτηση προτύπου a σε κείμενο x



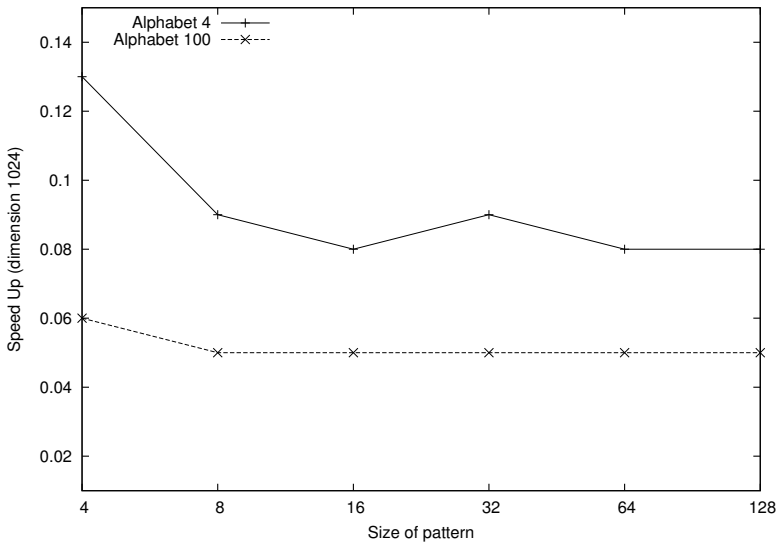
Προσεγγιστική Αναζήτηση προτύπου a σε κείμενο x

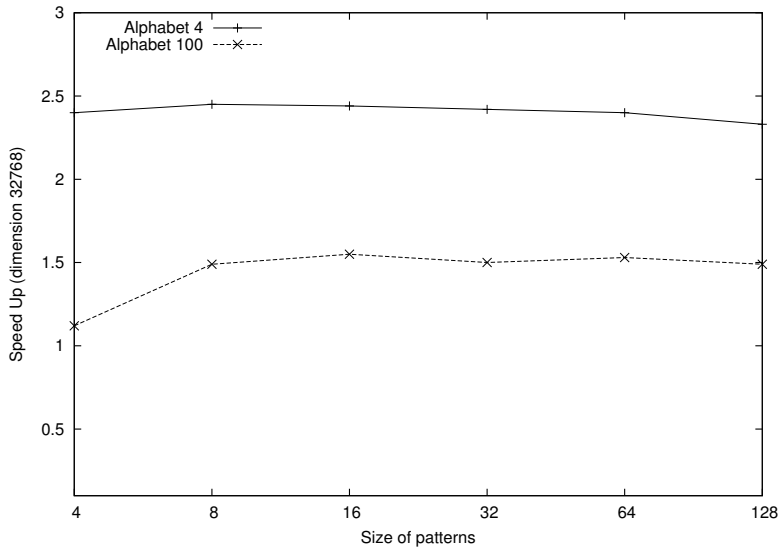


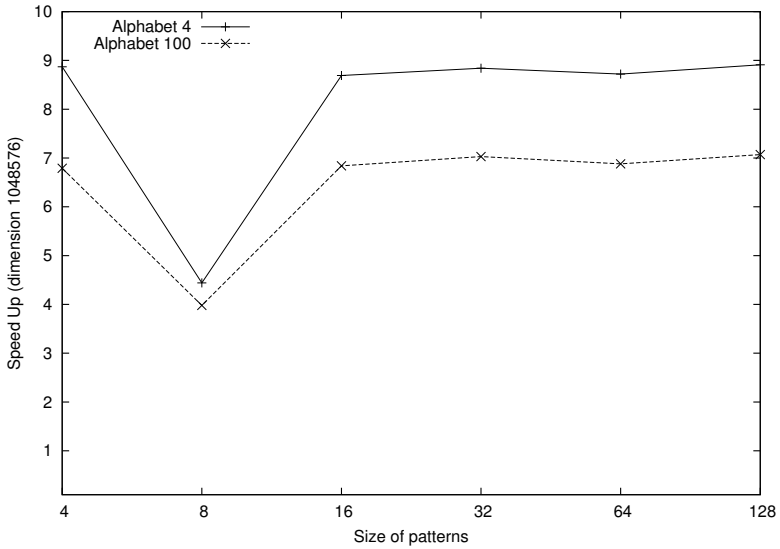


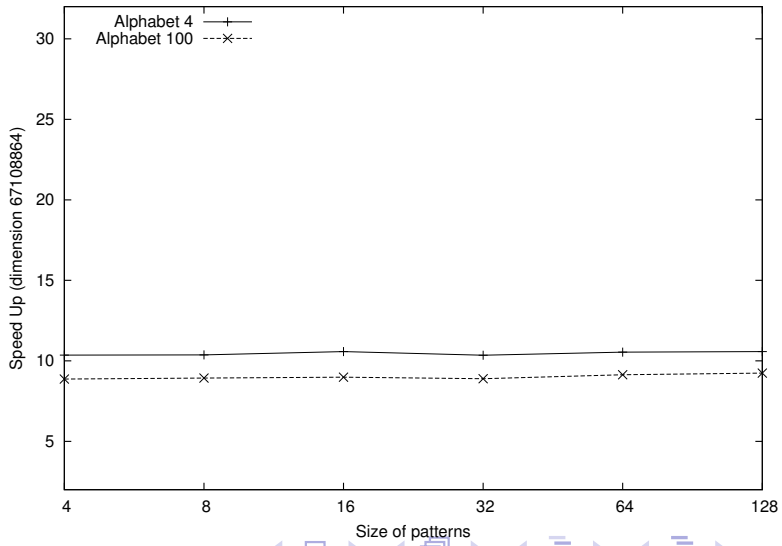
- Απεικονίζει την επιτάχυνση με βάση τα διαφορετικά μεγέθη κειμένου.
- Για μικρά μεγέθη κειμένου (έως 65536), παρατηρείται ότι όσο πιο μεγάλο είναι το μέγεθος του προτύπου τόσο καλύτερη επιτάχυνση έχει.
- Όσο το μέγεθος του κειμένου αυξάνεται παρατηρείται μία εξισορρόπηση στην επιτάχυνση η οποία παραμένει σχετικά σταθερή (περιορισμένους πόρους της κάρτας).

Αναζήτηση προτύπου a σε κείμενο x









Αναζήτηση πολλαπλών προτύπων

- Βασικό πρόβλημα στην επιστήμη της Πληροφορικής (ανάκτηση πληροφοριών, σε συστήματα ανίχνευσης επιθέσεων και στη βίο-πληροφορική).

Ορισμός

Δοθέντος ενός κειμένου εισόδου $T = t_1 t_2 ... t_n$, αποτελούμενο από ένα αλφάβητο Σ, και ενός πεπερασμένου συνόλου από r λέξεις-κλειδιά $P = \{p_1, p_2, ..., p_r\}$, όπου όλες έχουν σταθερό μήκος m χαρακτήρων και το συνολικό μέγεθος συμβολίζεται με $|P|$, η πολλαπλή αναζήτηση προτύπων είναι ο εντοπισμός όλων των τοποθεσιών i στο T, όπου υπάρχει εύρεση οποιασδήποτε λέξης.

Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

- Αλγόριθμος κατηγορίας κατακερματισμού (hashing).
- 1994 Sun Wu και τον Udi Manber και χρησιμοποιήθηκε στο εργαλείο glimpse.
- Σήμερα παραλλαγή του χρησιμοποιείται στο εργαλείο Snort (συστήματα ανίχνευσης επιθέσεων).
- Βασίζεται στον αλγόριθμο αναζήτησης προτύπου Boyer-Moore, παρακάμπτει μεγάλη ποσότητα του κειμένου κατά την αναζήτηση με μετατοπίσεις.
- Εξετάζει το κείμενο ανά μπλοκ χαρακτήρων μεγέθους B ($\log_{|\Sigma|} 2P$), γίνεται αύξηση του μεγέθους του αλφαβήτου.

Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

1 - Φάση προ-επεξεργασίας

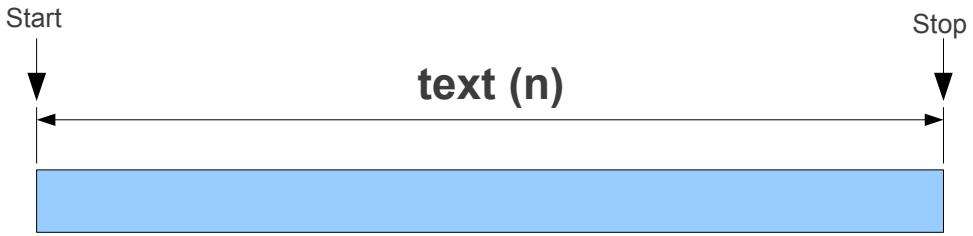
- Προ-επεξεργασία των προτύπων και δημιουργούνται 3 πίνακες.
- **SHIFT**: παρόμοιος με την τεχνική bad-character-shift του Boyer-Moore. (μέγιστη τιμή μετατόπισης $m - B + 1$)
- **HASH**: περιέχει κατακερματισμένες τιμές του επιθέματος B χαρακτήρων των προτύπων.
- **PREFIX**: περιέχει κατακερματισμένες τιμές του προθέματος B′ χαρακτήρων των προτύπων.

2 -Φάση αναζήτησης

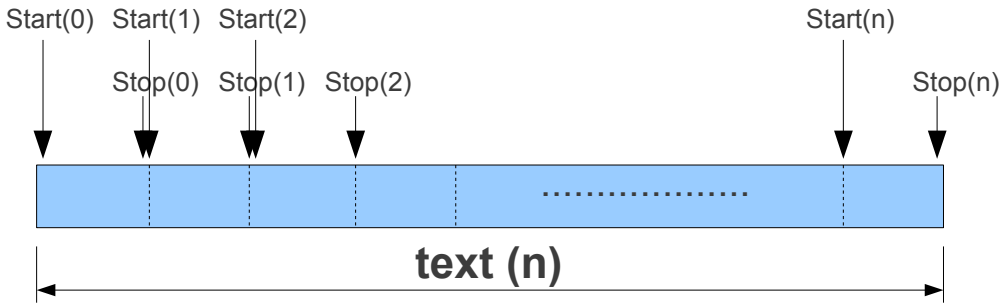
- Σαρώνεται το κείμενο έως ο δέικτης $T_p < T_{end}$.
- Υπολογίζεται η τιμή κατακερματισμού B χαρακτήρων από το επίθεμα του παραθύρου ($hash(sbc)$).
- Αν το SHIFT[hash(sbc)] == 0, γίνεται μετατόπιση κατά $T_p = T_p + SHIFT[hash(sbc)]$.
- Αν το $SHIFT[hash(sbc)] > 0$, υπάρχει πιθανό ταίριασμα και ελέγχεται η τιμή HASH(hash(sbc)) (δείκτη στην αρχή λίστας πρότυπων) και η PREFIX[(hash(sbc)].

Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

α) Σειριακή υλοποίηση



β) Παράλληλη υλοποίηση



Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

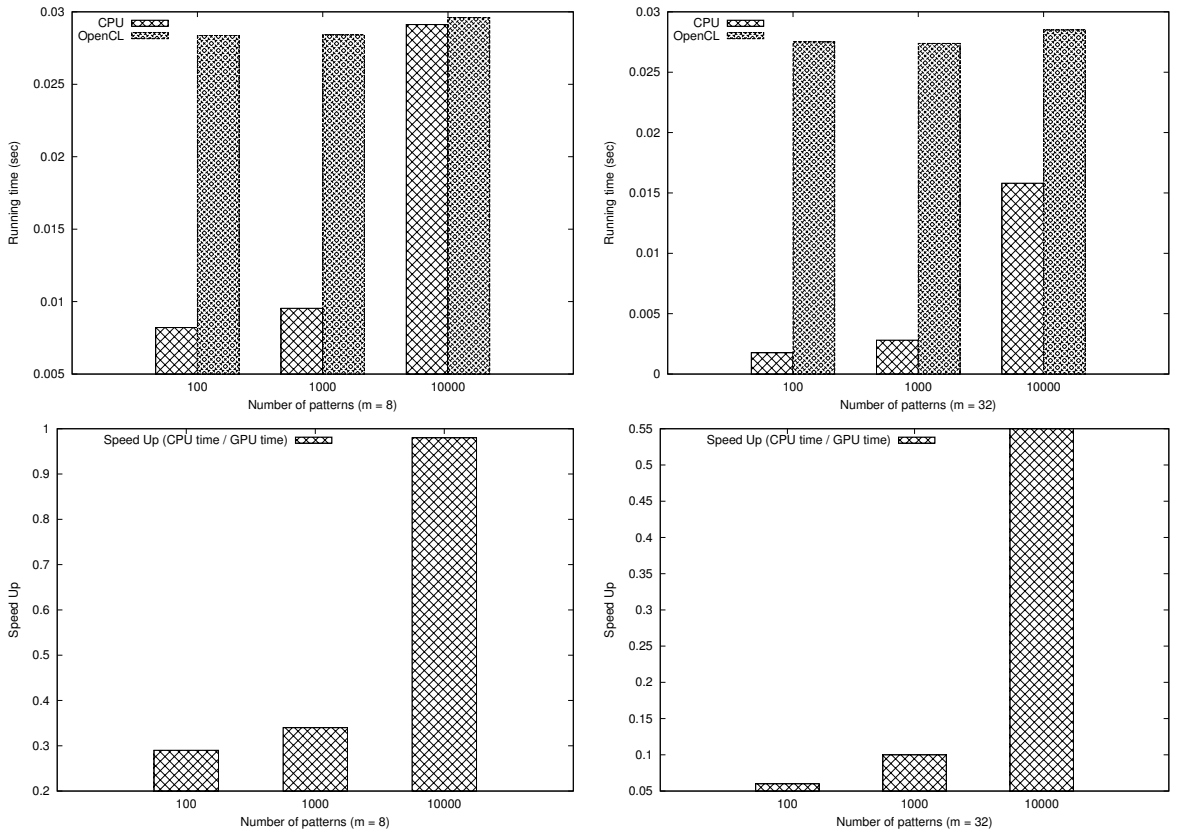
Μεθοδολογία πειραμάτων

Τα σύνολα δεδομένων που χρησιμοποιήθηκαν είναι:

- Φυσική γλώσσα αγγλικού αλφαβήτου ($n = 2.473.400$ και του αλφαβήτου είναι 94).
- Το γονιδίωμα του βακτήριου Εσερίχια κόλι (Escherichia coli) ($n = 4.638.690$ χαρακτήρες και αλφάβητο 4).
- FASTA Nucleidic Acid (FNA) του γονιδιώματος A-thaliana ($n = 118.100.062$ χαρακτήρες και αλφάβητο 4)
- Το FASTA Amino Acid (FAA) του γονιδιώματος A-thaliana ($n = 11.273.437$ χαρακτήρες και αλφάβητο 20)
- Τη βάση δεδομένων της ακολουθίας αμινοξέων SWISS-PROT ($n = 182.116.687$ και αλφάβητο 20)

Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

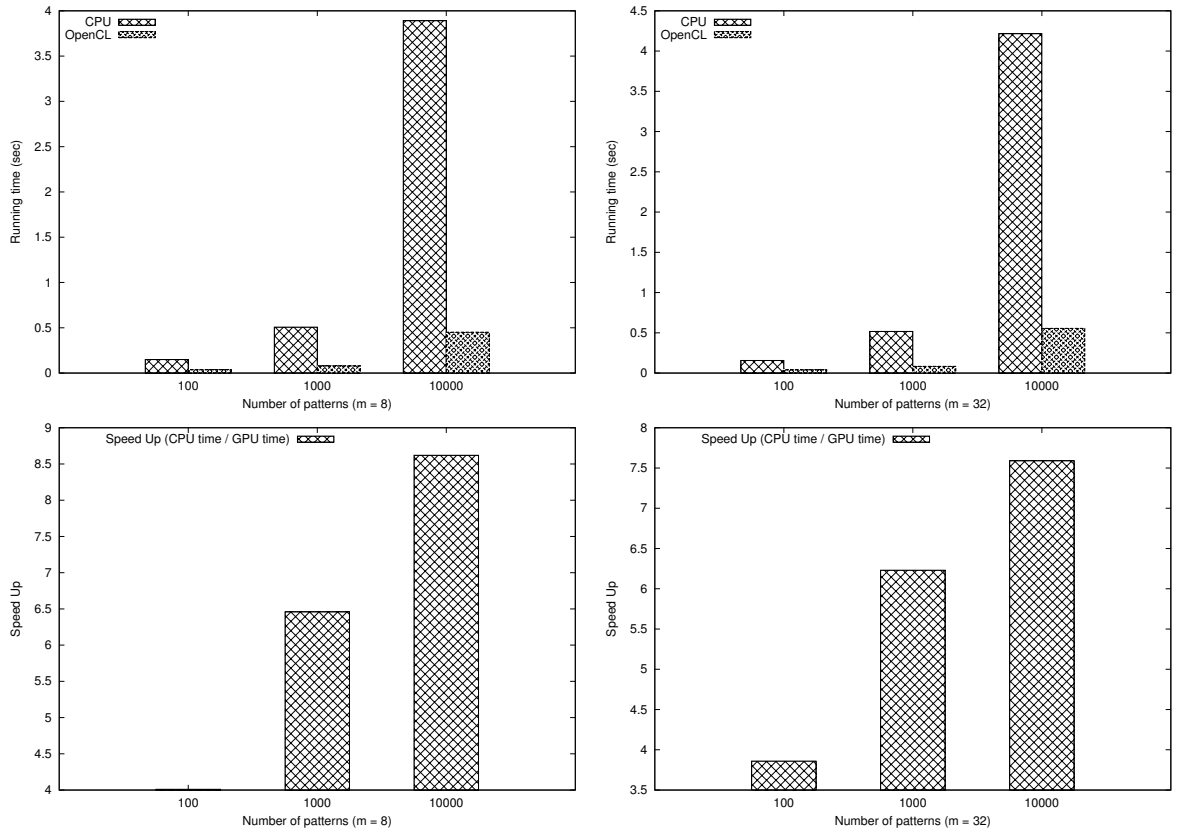
Αποτελέσματα πειράματος



Χρόνος εκτέλεσης και επιτάχυνση για κείμενο φυσικής γλώσσας (CIA World Factbook).

Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

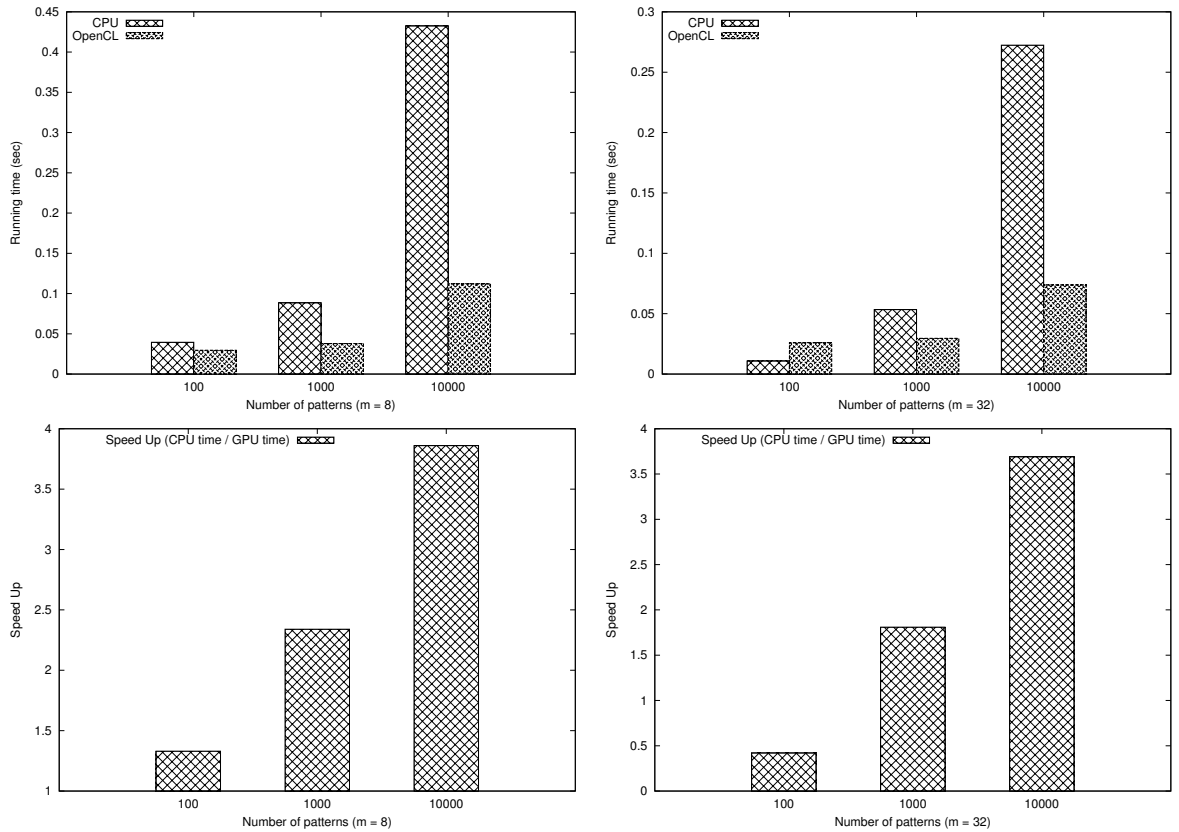
Αποτελέσματα πειράματος



Χρόνος εκτέλεσης και επιτάχυνση για κείμενο γονιδιώματος Εσερίχια κόλι (E.Coli).

Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

Αποτελέσματα πειράματος



Χρόνος εκτέλεσης και επιτάχυνση για κείμενο FASTA Amino Acid (FAA).

Αναζήτηση πολλαπλών προτύπων
Αλγόριθμος (Wu-Manber)

Αποτελέσματα πειράματος

- Η επιτάχυνση κυμαίνεται από 0,06 για φυσική γλώσσα και αριθμό προτύπων 100, έως 8,62 για κείμενο αλφαβήτου μεγέθους τέσσερα και αριθμό προτύπων 10.000.
- **Παράγοντας αλφαβήτου Σ**. Όσο μεγαλύτερο είναι το αλφάβητο τόσο η επιτάχυνση μειώνεται.
- **Παράγοντας μεγέθους κειμένου n**. Όσο αυξάνεται το μέγεθος τόσο αυξάνεται και η επιτάχυνση (0,98 $\rightarrow$ 3,07 για τετραπλάσιο μέγεθος φυσικού κειμένου).
- **Παράγοντας πλήθους προτύπων r**. Όσο αυξάνεται το πλήθος τόσο αυξάνεται και η επιτάχυνση.
- **Παράγοντας μεγέθους προτύπων m**. Δεν παίζει τόσο σημαντικό ρόλο στην επιτάχυνση κυρίως για μικρά αλφάβητα.

Περιεχόμενα

Συμπεράσματα

- Η παραλληλοποίηση είναι επιθυμητή μέθοδος όσο το πλήθος των πράξεων και των δεδομένων αυξάνεται.
- Τόσο τα πειράματα εκτέλεσης του αλγορίθμου Wu-Manber, όσο και τα παραδείγματα του πολλαπλασιασμού πίνακα με διάνυσμα εκτελέστηκαν με χρήση της global μνήμης του OpenCL και επιδέχονται περαιτέρω βελτίωση, με τη χρήση κατώτερων επιπέδων μνήμης.