ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Επιστημονικός Υπολογισμός Ι

Ενότητα **2** : Μοντέλα Υπολογισμού

Ευστράτιος Γαλλόπουλος

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο **«Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών»** έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σκοπός Ενότητας

- Μοντέλα υπολογισμού.
- Μοντέλο ιεραρχικής μνήμης.
- Μετρικές επίδοσης και απόδοσης προγραμμάτων στο μοντέλο ιεραρχικής μνήμης.

Περιεχόμενα

Παραδείγματα μίγματος εντολών μετροπρογραμμάτων (MK07)

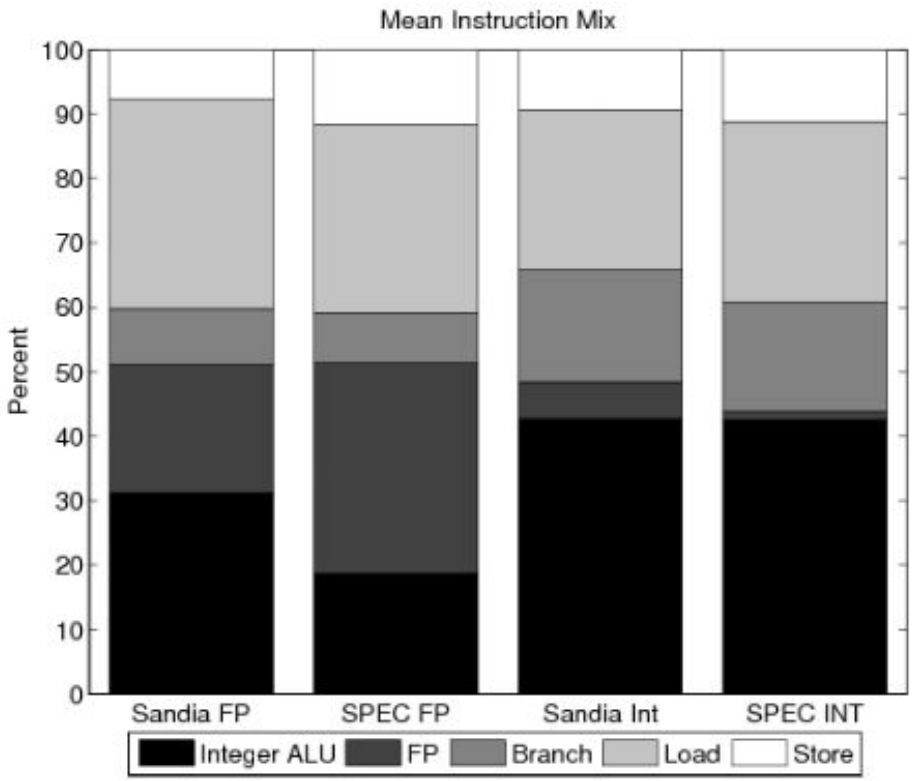


Fig. 2. Benchmark Suite Mean Instruction Mix

Μερικές οπτικοποιήσεις ≈ 1985 (hors-d'oeuvre)

4.1.4 MOD1d

Program MOD1D tries to generate memory bank conflicts in order to observe the effect on the performance of the machine. To this end the following kernel is performed:

```
DO 10 I = 1, INCR*1000, INCR
   S = S + A(I)*B(I)
10 CONTINUE
```

where INCR ranges from 1 to 32. The results are displayed in Figure 15.

INCR	Cray-2S Mflop/s	Cray Y-MP Mflop/s
1	114.9	236.0
2	51.3	228.3
3	114.6	234.1
4	32.4	142.0
5	114.9	229.9
6	51.5	229.6
7	114.9	233.4
8	32.6	117.2
9	114.9	233.2
10	51.8	228.9
11	114.0	233.3
12	32.0	142.2
13	114.9	234.4
14	51.6	228.4
15	114.9	234.6
16	32.6	62.0
17	114.9	236.1
18	51.3	229.1
19	114.6	229.2
20	32.4	143.1
21	114.9	231.9
22	51.5	230.1
23	114.9	234.6
24	32.6	116.3
25	114.9	234.5
26	51.6	229.9
27	114.9	235.6
28	32.6	142.2
29	114.9	233.8
30	51.6	229.1
31	114.9	235.0
32	32.6	32.4

Figure 15. Performance of the inner product kernel for different values of the increment.

Παράδειγμα

Κώδικας 1: pseudo-MATLAB Horner με LOAD/STORE

```
1  LOAD a(1:n+1), x
2  s = a(n+1);
3  for j=n:-1:1
4      s = s*x + a(j);
5  end;
6  STORE s;
```

Αν η cache έχει χωρητικότητα $\mathcal{K} = O(n)$, τότε $\Omega = 2n$, $\Phi = n + 3$.

$$\text{Συνοπτικά}: [\mathcal{K}, \Omega, \Phi] = [O(n), 2n, n + 3]$$

Συγκρίσεις

Στόχοι

1. υλοποιήσεις που ελαχιστοποιούν το χρόνο εκτέλεσης, επομένως ελαχιστοποιούν το Φ
2. υλοποιήσεις που πετυχαίνουν καλή ισορροπία (trade-off) μεταξύ ταχύτητας και κόστους
3. για δεδομένους πόρους, υλοποιήσεις που ελαχιστοποιούν το χρόνο εκτέλεσης

- η υλοποίηση φαίνεται ότι χρειάζεται $K = O(n)$
- μη πρακτικό όταν το K είναι συνάρτηση του n
- ... γιατί το μέγεθος του προβλήματος δεν είναι εκ των προτέρων γνωστό!

Παράδειγμα

Κώδικας 2: pseudo-MATLAB Horner με JIT LOAD/STORE

```
1  LOAD a(n+1), x;
2  s = a(n+1);
3  for j=n:-1:1
4      LOAD a(j);
5      s = s*x + a(j);
6  end;
7  STORE s;
```

Αν η cache έχει χωρητικότητα $\mathcal{K} = O(1)$, τότε $\Omega = 2n$, $\Phi = n + 3$.

$$\text{Συνοπτικά:} [K, \Omega, \Phi] = [O(1), 2n, n + 3]$$

Βασικοί υπολογισμοί με μητρώα

Τα πρώτα προβλήματα που θα εξετάσουμε είναι όλα της μορφής

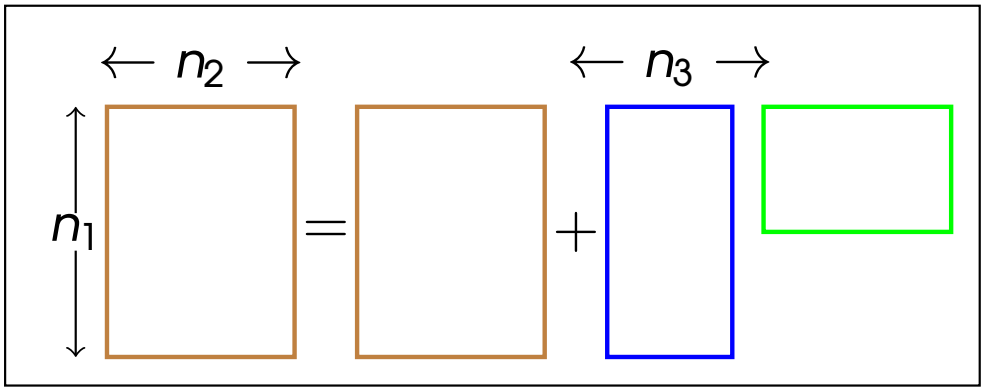
$$C \leftarrow C + AB, \quad C \in \mathbb{R}^{n_1 \times n_2}, A \in \mathbb{R}^{n_1 \times n_3}, B \in \mathbb{R}^{n_3 \times n_2}.$$

	$(n_1 > 1?n_2 > 1?n_3 > 1?)$	
DOT	$(0,0,1)$	εσωτερικό γινόμενο
DAXPY	$(1,0,0)$	διάνυσμα + βαθμωτός $\times$ διάνυσμα
	$(0,1,0)$	διάνυσμα + διάνυσμα $\times$ βαθμωτός
GER	$(1,1,0)$	ανανέωση 1ης τάξης
MV	$(1,0,1)$	διάνυσμα + μητρώο $\times$ διάνυσμα
	$(0,1,1)$	διάνυσμα + διάνυσμα $\times$ μητρώο
MM	$(1,1,1)$	μητρώο + μητρώο $\times$ μητρώο

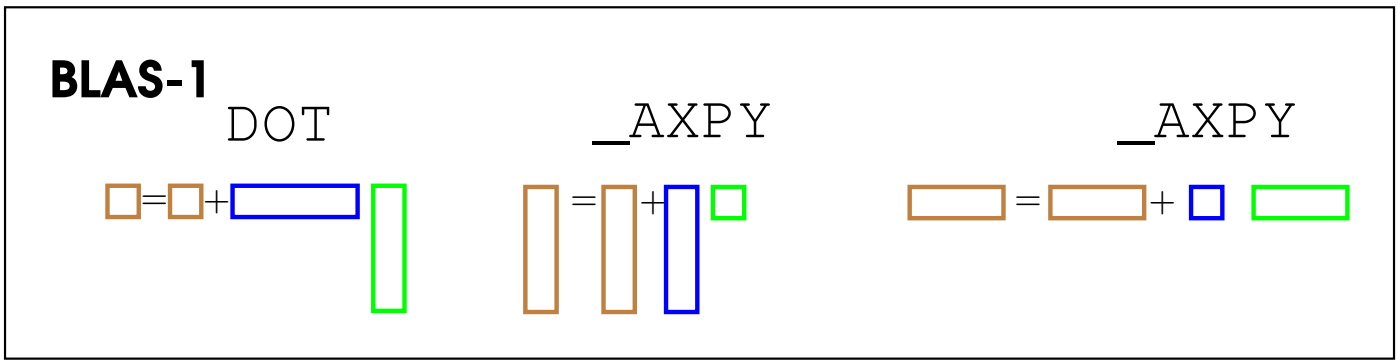
ΠΡΟΣΟΧΗ:

1. Προς το παρόν αναφερόμαστε σε **γενικά μητρώα χωρίς γνωστή δομή**
2. Στη MATLAB δίνεται πρόσβαση στις παραπάνω πράξεις μέσω τελεστών, π.χ. για βαθμωτό `t` και διανύσματα και μητρώα `a, b, C, D,E` εντολές όπως `a+t*b, t+a'*b,C+a*b', a+C*b, E+C*D`.

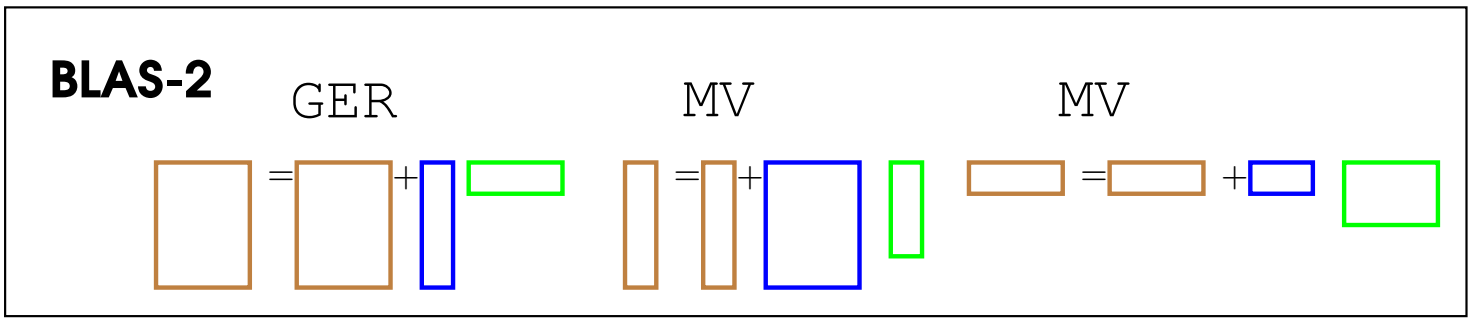
Template και ειδικές περιπτώσεις



BLAS-1



BLAS-2



BLAS: Basic Linear Algebra Subprograms

Από τη Wikipedia:

Basic Linear Algebra Subprograms (BLAS) is a de facto application programming interface standard for publishing libraries to perform basic linear algebra operations such as vector and matrix multiplication. They were first published in 1979, and are used to build larger packages such as LAPACK. Heavily used in high-performance computing, highly optimized implementations of the BLAS interface have been developed by hardware vendors ...

Τρία επίπεδα: Εν συντομία BLAS-1, BLAS-2, BLAS-3. Γενικά, αναφέρονται σε πράξεις μεταξύ αλγεβρικών αντικειμένων (μητρώων, διανυσμάτων) στα οποία μπορεί να υφίστανται 3 διαστάσεις συνολικά (οι βαθμωτοί δεν λαμβάνονται υπόψη)

- BLAS-1: 1 διάστ. $>$ 1, δηλ. πράξεις μεταξύ διανυσμάτων (π.χ. DOT, _AXPY).
- BLAS-2: 2 διαστ. $>$ 1, δηλ. πράξεις μεταξύ μητρώων, διανυσμάτων (π.χ. MV, ανανέωση τάξης-1)..
- BLAS-3: 3 διαστ. $>$ 1, δηλ. πράξεις μεταξύ μητρώων (π.χ. MM).

BLAS: Basic Linear Algebra Subprograms

- API για βασικές πράξεις της ΑΓΑ. Περιγράφονται η ονοματολογία, ο τρόπος κλήσης και οι πράξεις, όχι όμως η υλοποίηση.
- Συνοπτική παρουσίαση http://www.netlib.org/lapack/lug/node145.html
- Υλοποιήσεις:
  - Κώδικες αναφοράς Fortran http://netlib.org/blas/
  - ATLAS Automaticaly Tuned Linear Algebra Subroutines για C, Fortran
  - GotoBLAS
  - MKL BLAS
  - OpenBLAS
  - και πολλές άλλες ...

BLAS-1: Δείκτης τοπικότητας Ι

Κώδικας 3: Pseudo-MATLAB BLAS daxpy with L/S

```
1  LOAD(a, x(1:n), y(1:n));
2  for j=1:n
3      y(j) = y(j) + a*x(j);
4  end
5  STORE(y(1:n));
```

$$[K, \Omega, \Phi_{\text{min}}] = [2n + O(1), 2n, 3n + 1] \Rightarrow \mu_{\text{min}} = \frac{3}{2} + \frac{1}{2n}$$

Κώδικας 4: Pseudo-MATLAB BLAS ddot with L/S

```
1  LOAD(s, x(1:n), y(1:n));
2  for j=1:n
3      s = s + x(j)*y(j);
4  end
5  STORE(s);
```

$$[K, \Omega, \Phi_{\text{min}}] = [2n + O(1), 2n, 2n + 2] \Rightarrow \mu_{\text{min}} = 1 + \frac{1}{n}$$

BLAS-1: Δείκτης τοπικότητας II

Κώδικας 5: Pseudo-MATLAB BLAS daxpy with JIT L/S

```
1  LOAD(a);
2  for j=1:n
3      LOAD(x(j),y(j));
4      y(j) = y(j) + a*x(j);
5      STORE(y(j));
6  end
```

$$[\mathcal{K}, \Omega, \Phi] = [O(1), 2n, 3n+1] \Rightarrow \mu_{\min} = \frac{3}{2} + \frac{1}{2n}$$

Σημείωση: Χρησιμοποιούμε το αρκτικόλεξο JIT εννοώντας just in time, δηλ. η μεταφορά εκτελείται ακριβώς πριν χρειαστεί το δεδομένο.

ΠΡΟΣΟΧΗ: Η υλοποίηση επιτυγχάνει $\Phi = \Phi_{\min}$ μόνο με $\mathcal{K} = O(1)$ cache!

Τι ενδιαφέρει

Δίδονται ένα ή περισσότερα προγράμματα ή 'ψευδο-κώδικες':

Κώδικας 6: Pseudo-MATLAB

```
1 statement_First;
2 statement_Second;
3 ...
4 statement_Last;
```

Τυπικά ζητούμενα από υλοποιήσεις εντός του μοντέλου

1. Οι τιμές Ω, Φ_{min}, μ_{min}
2. Δοθέντος K, στρατηγική ένθεση εντολών LOAD, STORE για μείωση των Φ, μ.
3. Νέα υλοποίηση που επιτυγχάνει (ακόμα) μικρότερα Φ, μ, όσο γίνεται πλησιέστερα στο Φ_{min}.
4. Συστηματική αξιολόγηση/σύγκριση υλοποιήσεων.
5. Θα θέλαμε οι βελτιώσεις να γίνονται αισθητές και στις υλοποιήσεις σε υπάρχοντα υπολογιστικά συστήματα.

Συγκρίσεις

- η δεύτερη υλοποίηση πετυχαίνει την ίδια πολυπλοκότητα πράξεων και μεταφορών με $\mathcal{K} = O(1)$ αντί $O(n)$
- μικρότερη απαίτηση σε πόρους (μικρότερο κόστος)

Θέματα

1. υπολογισμός Φ_{min} του αλγορίθμου (πώς;) ... ΕΥΚΟΛΟ
2. σχεδιασμός υλοποίησης που ελαχιστοποιεί το $\Phi \geq \Phi_{\text{min}}$ ... ΔΥΣΚΟΛΟΤΕΡΟ

Κώδικας 7: Υλοποίηση αλγορίθμου με $\Phi = \Phi_{\text{min}}$

```
1 LOAD all_data_in_cache; %Phase_1
2 COMPUTE all;                    % Phase 2
3 STORE results_in_memory; % Phase 3
```

Η υλοποίηση προυποθέτει απεριόριστη cache αλλά δείχνει ότι το Φ_{min} είναι το πλήθος των (χρήσιμων) δεδομένων εισόδου που φορτώνονται στο LOAD και εξόδου που αποθηκεύονται στο STORE και είναι εύκολο να την υπολογίσετε!

Κοινά χαρακτηριστικά DOT, _AXPY

- $n_i, n_j = 1, n_k > 1$ για $i, j, k \in \{1, 2, 3\}$
- $\Omega = O(n)$, $\Phi_{\text{min}} = O(n)$, $\mu_{\text{min}} = O(1)$
- Πολυπλοκότητες γραμμικές στην κυρίαρχη διάσταση
- Μη αποδοτική υλοποίηση σε ιεραρχική μνήμη
- Level-1 Basic Linear Algebra Subprograms (BLAS-1)

Βασικές πράξεις γραμμικής άλγεβρας 1ου επιπέδου

BLAS-2: Ανανέωση 1ης τάξης (rank-1 update) I

Κώδικας 8: Pseudo-MATLAB BLAS-2 with L/S

```
% Par'adeigma rank-1 update
% C <- C + a*b'
LOAD(C,a,b)
for j = 1:n2
    for i=1:n1
        C(i,j) = C(i,j) + a(i)*b(j);
    end
end
STORE(C)
```

$$[\mathcal{K}, \Omega, \Phi] = [O(n_1 n_2), 2n_1 n_2, 2n_1 n_2 + n_1 + n_2] \Rightarrow \mu_{\min} = 1 + \frac{1}{2n_1} + \frac{1}{2n_2}.$$

BLAS-2: Ανανέωση 1ης τάξης (rank-1 update) II

Κώδικας 9: Pseudo-MATLAB BLAS-2 with L/S

```
1  % Par'adeigma rank-1 update
2  % C <- C + a*b'
3  for j = 1:n2
4      LOAD(b(j))
5      for i=1:n1
6          LOAD(C(i,j), a(i))
7          C(i,j) = C(i,j) + a(i)*b(j)
8          STORE(C(i,j))
9      end
10 end
```

$$[\mathcal{K}, \Omega, \Phi] = [O(1), 2n_1n_2, 3n_1n_2 + n_2] \Rightarrow \mu = \frac{3}{2} + \frac{1}{2n_1}$$

BLAS-2: Ανανέωση 1ης τάξης (rank-1 update) III

Κώδικας 10: Pseudo-MATLAB BLAS-2 with L/S

```
1  % Par'adeigma rank-1 update
2  % C <- C + a*b'
3  LOAD(a(1:n1))
4  for j = 1:n2
5      LOAD(b(j))
6      for i=1:n1
7          LOAD(C(i,j))
8          C(i,j) = C(i,j) + a(i)*b(j)
9          STORE(C(i,j))
10     end
11 end
```

$$[\mathcal{K}, \Omega, \Phi] = [O(n_1), 2n_1n_2, \underbrace{2n_1n_2 + n_1 + n_2}_{\Phi_{\min}}] \Rightarrow \mu = 1 + \frac{1}{2n_1} + \frac{1}{2n_2} = \mu_{\min}$$

Παρατηρήσεις

Πλεονεκτήματα

- Πετύχαμε υλοποίηση με μ_{min} και $\mathcal{K} = O(n_1) < O(n_1 n_2)$
- Θα μπορούσαμε το ίδιο με $\mathcal{K} = O(n_2) < O(n_1 n_2)$
- Επομένως μπορούμε να πετύχουμε μ_{min} με $\mathcal{K} = O(\min(n_1, n_2))$.

Μειονεκτήματα

Χρειάζεται κρυφή μνήμη μεγέθους $O(n_1)$ ή $O(n_2)$.

ΠΡΟΚΛΗΣΗ

Να ξεπεράσουμε την (ανεπιθύμητη) εξάρτηση του μεγέθους της cache από τις διαστάσεις του προβλήματος;

Τεμαχισμός (σε πλοκάδες)

$$\begin{pmatrix} C_{11} & \cdots & C_{1,k_2} \\ C_{21} & \ddots & \vdots \\ \vdots & C_{I,J} & \vdots \\ \vdots & \ddots & \vdots \\ C_{k_1,1} & \cdots & C_{k_1,k_2} \end{pmatrix} = \begin{pmatrix} C_{11} & \cdots & C_{1,k_2} \\ C_{21} & \ddots & \vdots \\ \vdots & C_{I,J} & \vdots \\ \vdots & \ddots & \vdots \\ C_{k_1,1} & \cdots & C_{k_1,k_2} \end{pmatrix} + \begin{pmatrix} a_1 \\ \vdots \\ a_I \\ \vdots \\ a_{k_1} \end{pmatrix} \begin{pmatrix} b_1 & \cdots & b_{k_2} \end{pmatrix}$$

Αν υποθέσουμε ότι για $j = 1, 2, 3$:

$$n_j = \underbrace{(\text{πλοκάδες στη διάσταση } j)}_{k_j} \underbrace{(\text{μέγεθος πλοκάδας στη διάσταση } j)}_{m_j}$$

Με μαθηματική γραφή (το b_J είναι γραμμή), για $I = 1, \ldots, k_1$; $J = 1, \ldots, k_2$:

$$\boxed{C_{IJ} = C_{IJ} + a_I b_J, \quad C_{IJ} \in \mathbb{R}^{m_1 \times m_2}, a_I \in \mathbb{R}^{m_1}, b_j \in \mathbb{R}^{m_2}}$$

Στην πράξη

Με τις παραπάνω διαστάσεις, έχουμε τις ακολουθες αντιστοιχίες μεταξύ αλγεβρικών συμβόλων για υπομητρώα και στοιχεία των πινάκων που αποθηκεύονται τα `C, a, b`:

Υπομητρώο/πλοκάδα	αντιστοιχεί στα στοιχεία	μέγεθος
C_{IJ},	`C((I-1)*m1+1:I*m1,(J-1)*m2+1: J*m2)`	$m_1 \times m_2$
a_I	`a((I-1)*m1+1:I*m1)`	m_1
b_J	`b((J-1)*m2+1:J*m2)`	m_2

Παρατηρήσεις:

1. Ο τεμαχισμός μπορεί να είναι ανισομερής, αρκεί να είναι συμβατές οι διαστάσεις
2. Χρησιμοποιούμε μαθηματικά σύμβολα και υποδείκτες για τα μαθηματικά
3. Χρησιμοποιούμε αντίστοιχα σύμβολα σε διαφορετική γραμματοσειρά αλλά με παρενθέσεις και χωρίς υποδείκτες για τα δεδομένα

BLAS-2: Ανανέωση 1ης τάξης (rank-1 update) με τεμαχισμό σε πλοκάδες I

Κώδικας 11: Pseudo-MATLAB BLAS-2

```
% Par'adeigma rank-1 update
% C <- C + a*b'
for J=1:k2
    for I=1:k1
% Ulopo'ihsh thc anan'ewshc sthn plok'ada
        C((I-1)*m1+1:I*m1,(J-1)*m2+1: J*m2) = ...
                C((I-1)*m1+1:I*m1,(J-1)*m2+1: J*m2) +...
                    a((I-1)*m1+1:I*m1)*(b((J-1)*m2+1:J*m2))
    end
end
```

BLAS-2: Ανανέωση 1ης τάξης (rank-1 update) με τεμαχισμό σε πλοκάδες II

Κώδικας 12: Pseudo-MATLAB BLAS-2 with L/S

```
1  for I=1:k1
2      LOAD(a((I-1)*m1+1:I*m1));
3      for J=1:k2
4          C((I-1)*m1+1:I*m1,(J-1)*m2+1: J*m2) = ...
5                  C((I-1)*m1+1:I*m1,(J-1)*m2+1: J*m2) +...
6                      a((I-1)*m1+1:I*m1)*b((J-1)*m2+1:J*m2);
7      end
8  end
```

$$\begin{aligned} \Phi &= ((2m_1m_2 + m_2)k_2 + m_1)k_1 = 2n_1n_2 + n_1 + n_2k_1 \\ \mu &= 1 + \frac{1}{2m_1} + \frac{1}{2n_2} \geq \mu_{\min}. \end{aligned}$$

Το πρόβλημα

Δίνονται $\mathcal{K}, n_1, n_2$ και αναζητούμε τεμαχισμό που ελαχιστοποιεί το μ.

- Προσέξτε: μεγαλύτερη πλοκάδα του a στην cache συνεπάγεται λιγότερες επαναλήψεις.
- ... είναι λογικό να επιλέξουμε το μέγιστο m_1 που επιτρέπεται από την cache, άρα

$$\min_{m_1 \leq \mathcal{K}} \mu = 1 + \frac{1}{2\mathcal{K}} + \frac{1}{2n_2}$$

- Αν $n_1 \leq \mathcal{K}$ τότε θέτουμε $m_1 = n_1$ και κατά συνέπεια $\min_{m_1 \leq \mathcal{K}} \mu = \mu_{\min}$.

Τροποποίηση

Ο κώδικας χρησιμοποιεί κλήση σε «συνάρτηση» `ger` για ανανέωση τάξης-1 μεγέθους $m_1 \times m_2$:

Κώδικας 13: Pseudo-MATLAB BLAS-2 block with L/S

```
1  for J=1:k2
2      for I=1:k1
3          LOAD(a((I-1)*m1+1:I*m1));
4          C((I-1)*m1+1:I*m1,(J-1)*m2+1: J*m2) = ...
5              ger(C((I-1)*m1+1:I*m1,(J-1)*m2+1: J*m2),...
6                a((I-1)*m1+1:I*m1),b((J-1)*m2+1:J*m2));
7      end
8  end
```

Συνάρτηση ger

Κώδικας 14: Pseudo-MATLAB BLAS-2 block with L/S

```
1 function C = ger(C,a,b);
2 for j = 1:n2
3     LOAD(b(j)) % upoj'etoume a in cache
4     for i=1:n1
5         LOAD(C(i,j))
6         C(i,j) = C(i,j) + a(i)*b(j)
7         STORE(C(i,j))
8     end
9 end
```

Παρατηρήσεις

Ο τεμαχισμός σε πλοκάδες (πλοκαδοποίηση - blocking) είναι κλειδί για την συγγραφή αποδοτικού κώδικα για ιεραρχική μνήμη.

- Ο καλύτερος τεμαχισμός εξαρτάται από το μέγεθος της κρυφής μνήμης και των καταχωρητώ (cache aware).
- Η ελαχιστοποίηση βασίστηκε σε μια βέλτιστη υλοποίηση όμοιου αλλά μικρότερου προβλήματος με διαστάσεις που επιλέξαμε εμείς.
- Για να διερευνησουμε την καλύτερη υλοποίηση πρέπει να εξετάσουμε και εναλλακτικές εμφωλεύσεις των βρόχων!

Βιβλιογραφία I

Richard C. Murphy and Peter M. Kogge.

On the memory access patterns of supercomputer applications: Benchmark selection and its implications.

IEEE Transactions on Computers, 56(7):937--945, 2007.

Ε. Γαλλόπουλος.

Επιστημονικός Υπολογισμός Ι.

Πανεπιστήμιο Πατρών, 2008.

Χρήση Έργου Τρίτων Ι

1. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.5233&rep=rep1&type=pdf (βλ. σελ 5-6)

Σημείωμα Αναφοράς

``Επιστημονικός Υπολογισμός Ι'', Έκδοση: 1.0, Πάτρα 2013-2014.

Διαθέσιμο από τη δικτυακή διεύθυνση: https://eclass.upatras.gr/courses/CEID1096/

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης