ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ: ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΠΑΡΑΛΛΗΛΕΣ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΥΠΟΛΟΓΙΣΜΟΥ ΓΙΑ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΕΞΕΤΑΣΕΙΣ ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2022-23

16 Ιουνίου 2023

ΔΙΑΡΚΕΙΑ: 2 ½ ώρες

ΟΝΟΜΑ: [redacted]

Α.Μ. [redacted]

Θέμα 1 (5μ): *Εκπαιδεύετε ένα βαθύ νευρωνικό δίκτυο για αναγνώριση εικόνων. Για τα τέσσερα σενάρια της αριστερής στήλης, θα χρειαστεί παράλληλη επεξεργασία. Με ποιον τρόπο θα την υλοποιήσετε σε κάθε σενάριο; Μπορείτε να επιλέξετε παραπάνω από μία απαντήσεις από τη στήλη δεξιά. Αιτιολογήστε τις απαντήσεις σας.*

1) Θέλω να μειώσω το χρόνο εκπαίδευσης ενός νευρωνικού δικτύου	*α) Θα χρησιμοποιήσω περισσότερους πυρήνες (με κοινή μνήμη)*
2) Θέλω να εκπαιδεύσω ένα νευρωνικό δίκτυο με περισσότερα δεδομένα	*β) Θα χρησιμοποιήσω περισσότερους επεξεργαστές (με κατανεμημένη μνήμη)*
3) Θέλω να επιλέξω τις σωστές υπερπαραμέτρους για το νευρωνικό μου δίκτυο	*γ) Θα χρησιμοποιήσω επιταχυντές*
4) Θέλω να κάνω γρήγορο inference	

1) α, ~~β~~, γ, ανάλογα την περίπτωση.

• Το α θα βοηθήσει αν δεν χρησιμοποιούμε ήδη τόσους πυρήνες για να έχουμε φτάσει στο όριο κλιμακωσιμότητας → νομίζω τυπικά δεν με επιταχύνσει σε σχέση με κάτι άλλο που φτάσει το ίδιο

~~• Το β θα βοηθήσει, και είναι ίσως και η μόνη επιλογή για μεγάλα δίκτυα~~

• Το γ βοηθάει σε κάθε κλασσικό νευρωνικό, που κάνει βαριά χρήση πολλαπλασιασμού πινάκων στην εκπαίδευσή του.

2) β. Η μόνη επιλογή αν τα δεδομένα δεν χωράνε σε μη κατανεμημένο σύστημα

3) β. γιατί ο κάθε επεξεργαστής μπορεί να δοκιμάσει τιμές ανεξάρτητα.

4) γ. Τα εξειδικευμένα συστήματα (π.χ. FPGA) είναι ξεκάθαρα η καλύτερη επιλογή για γρήγορο inference

Θέμα 2 (6μ: 3 + 3): *α) Αναφέρετε ένα πλεονέκτημα και ένα μειονέκτημα γιά τις πιο διαδεδομένες τεχνικές παραλληλισμού ("Data" και "Model" parallelism) για την εκπαίδευση βαθιών νευρωνικών δικτύων*

Data	Model
+ ανεξάρτητες δουλειές → μικρότερη επικοινωνία κόμβων	+ δυνατότητα εκπαίδευσης μεγάλων δικτύων.
ων μοντέλων − αντίγραφο* για κάθε κόμβο → μεγάλο redundancy	− μεγάλη επικοινωνία και για backpropagation όλο μαζί

β) Παραλληλοποιώ την εκπαίδευση ενός νευρωνικού δικτύου με N νευρώνες. Χρησιμοποιώ p επεξεργαστές και το σύνολο εκπαίδευσης περιλαμβάνει M δεδομένα. Πόσους υπολογισμούς αναλαμβάνει κάθε επεξεργαστής με καθεμία από τις τεχνικές του ερωτήματος (α); Πόσα δεδομένα;

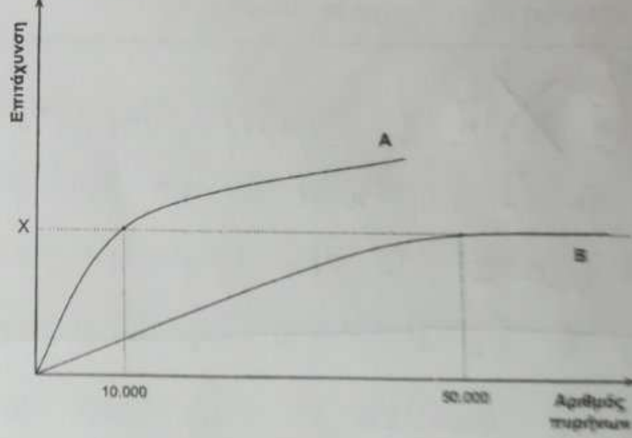
Data: N υπολογισμούς, M/p δεδομένα (τάξη μεγέθους πάντα)

Model: N/p υπολογισμούς, M δεδομένα

Θέμα 3 (3μ): *Για καθεμία από τις παρακάτω προτάσεις, σημειώστε ποια αναφέρεται σε αρχιτεκτονικές κοινής μνήμης (shared - S) και ποια αναφέρεται σε αρχιτεκτονικές κατανεμημένης μνήμης (distributed - D).*

Η κλιμακωσιμότητα της αρχιτεκτονικής είναι περιορισμένη.	S
Ο προγραμματιστής έχει την ευθύνη της ανταλλαγής των δεδομένων.	S
Πολλοί εργάτες ενός προγράμματος έχουν πρόσβαση στα ίδια δεδομένα.	S
Η ορθή εκτέλεση ενός παράλληλου προγράμματος σ'αυτή την αρχιτεκτονική μπορεί να οδηγήσει σε ανάγκη συγχρονισμού.	S
Η ορθή εκτέλεση ενός παράλληλου προγράμματος σ'αυτή την αρχιτεκτονική μπορεί να οδηγήσει σε ανάγκη επικοινωνίας.	D
Κάθε εργάτης ενός παράλληλου προγράμματος έχει πρόσβαση μόνο σε δεδομένα που βρίσκονται στη δική του μνήμη.	D

Θέμα 4 (4μ): *Υποθέστε ότι έχετε στη διάθεσή σας τους δύο από τους ταχύτερους υπερυπολογιστές στον κόσμο για να εκτελέσετε το πρόγραμμά σας. Σας δίνεται το παρακάτω διάγραμμα που περιγράφει την επιτάχυνση που μπορεί να επιτύχει κάθε υπερυπολογιστής για το πρόγραμμά σας συναρτήσει του αριθμού των πυρήνων του.*



Αν οι πυρήνες του υπερυπολογιστή A καταναλώνουν 6 φορές περισσότερη ισχύ από αυτούς του B και σας έχει ζητηθεί να επιταχύνετε το πρόγραμμά X φορές με τη μικρότερη δυνατή κατανάλωση ισχύος, σε ποιο σύστημα είναι προτιμότερο να το εκτελέσετε;

Ισχύς A: $10.000 \cdot P_A$

Ισχύς B: $50.000 \cdot P_B$

$P_A = 6 P_B$

Ισχύς A $= 60.000\, P_B > 50.000\, P_B =$ Ισχύς B.

Άρα προτιμότερο είναι το σύστημα B

Θέμα 5 (3μ): *Δώστε 2 βασικά χαρακτηριστικά τα οποία διαφοροποιούν την αρχιτεκτονική μιας GPU από αυτή μιας CPU.*

1. CPU: λίγοι και πιο ισχυροί πυρήνες
   GPU: πάρα πολλοί και απλοί πυρήνες.
2. CPU RAM: μεγάλη, μικρό bandwidth
   GPU RAM: μικρή, μεγάλο bandwidth.

Θέμα 6 (3μ): *Εξηγήστε γιατί οι σύγχρονες GPUs είναι κατάλληλες για την εκπαίδευση νευρωνικών δικτύων.*

Οι GPUs είναι πολύ κατάλληλες για tasks όπου:
- η branching λογική δεν είναι περίπλοκη.
- οι υπολογισμοί είναι πολύ "streamlined".
- οι υπολογισμοί ανά byte δεδομένων (O.I.) είναι πολλοί

→ και τα ισχυρά τους πολλούς πυρήνες που κατέχουν μεγάλο ποσοστό των συνολικών τρανζίστορ στην εκπαίδευση νευρωνικών

Θέμα 7 (3μ): *Πώς ορίζεται η επιτάχυνση (speedup) ενός προγράμματος; Τί είναι γραμμική (linear) και τι υπεργραμμική (superlinear) επιτάχυνση;*

Χρόνος εκτέλεσης του σειριακού προγράμματος (σειριακού)
= Χρόνος εκτέλεσης του (βέλτιστου παράλληλου) προγράμματος σε P επεξεργαστές

Επιτάχυνση $S = \frac{T_S}{T_P}$ ($\leq P$ θεωρητικά)

Γραμμική επιτάχυνση: $S = P$

Υπεργραμμική επιτάχυνση: $S > P$ (να αμφισβητείται με προσοχή)

Θέμα 8 (4μ): *Χρησιμοποιώντας το Νόμο του Amdahl, δείξτε τι είναι αποτελεσματικότερο:*
- *να κάνουμε 20% των εντολών ενός προγράμματος να εκτελεστούν 80% πιο γρήγορα ή*
- *να κάνουμε 80% των εντολών ενός προγράμματος να εκτελεστούν 20% πιο γρήγορα.*

Θεωρητική επιτάχυνση σύμφωνα με Amdahl.

$$S_1 = \frac{1}{0.8 + \frac{0.2}{1.8}} = \frac{1}{0.8 + \frac{1}{9}} = \frac{9}{7.2+1} = \frac{9}{8.2} = \frac{45}{41}$$

$$S_2 = \frac{1}{0.2 + \frac{0.8}{1.2}} = \frac{1}{0.2 + \frac{2}{3}} = \frac{3}{0.6+2} = \frac{3}{2.6} = \frac{15}{13}$$

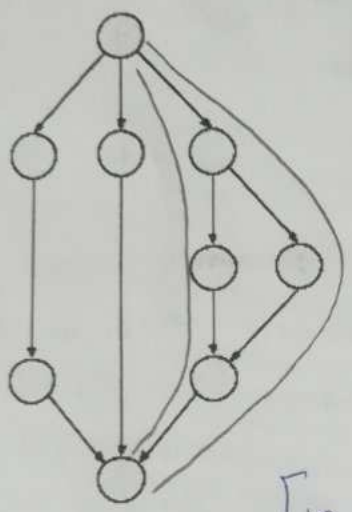
$S_2 > S_1$, άρα είναι αποτελεσματικότερο να κάνουμε το 80% να

Θέμα 2 ...

Θέμα 9 (3μ): *Αναφέρετε 3 λόγους για τους οποίους ένα πρόγραμμα μπορεί να μην κλιμακώνει.*

1. Πολλές εξαρτήσεις δεδομένων (δεν είναι παραλληλοποιήσιμο)
2. Overhead της παραλληλοποίησης είναι μεγάλο.
3. Απαιτείται πολλή επικοινωνία μεταξύ των επεξεργαστικών κόμβων.

Θέμα 10 (5μ): *Υπολογίστε τη μέγιστη επιτάχυνση για το γράφο εξαρτήσεων (task graph) που σας δίνεται στη συνέχεια, αν υποθέσουμε πως έχετε άπειρους επεξεργαστές.*



Το critical path (μεγαλύτερο μονοπάτι στον γράφο εξαρτήσεων) είναι το επισημασμένο (οποιοδήποτε από τα 2)

Έχει 5 κόμβους

Άρα, $T_\infty = 5$

Για 1 επεξεργαστή, πρέπει να περάσει όλους τους κόμβους, που είναι 9 $\Rightarrow T_1 = 9$

Άρα, η μέγιστη επιτάχυνση είναι $\frac{T_1}{T_\infty} = \frac{9}{5} = 1.8$

Θέμα 11 (8μ: 4 + 4):

α) *Ο παρακάτω κώδικας υπολογίζει το γινόμενο Hadamard δύο δισδιάστατων πινάκων A και B και αποθηκεύει το αποτέλεσμα σε έναν τρίτο πίνακα C. Παραλληλοποιήστε τον παρακάτω κώδικα με τη χρήση OpenMP.*

```
#pragma omp parallel for collapse(2)
for (i = 0 ; i < N ; i++)
    for (j = 0 ; j < M ; j++)
        C[i][j] = A[i][j] * B[i][j];
```

β) *Ο παρακάτω κώδικας υπολογίζει το π. Κάποιος προσπάθησε να τον παραλληλοποιήσει με χρήση OpenMP, αλλά το αποτέλεσμα δεν είναι σωστό. Βρείτε το λάθος και δώστε μία πιθανή λύση.*

```
int i;
int steps = 1000;
```

```
double x;
double pi, sum = 0.0;
double step = 1.0/(double) steps;
sum = 0.0;

#pragma omp parallel for reduction(+:sum)
for (i=0; i < steps; i++) {
    x = (i+0.5)*step;
    sum = sum + 4.0 / (1.0+x*x);
}

pi = step * sum;
```

Η μεταβλητή step είναι shared, τα threads των καιρών mutate, άρα έχουμε race condition.
Μια πιθανή λύση γράφεται παραπάνω.

Θέμα 12 (3μ): *Ο παρακάτω κώδικας OpenMP, όταν εκτελεστεί, θα εκτυπώσει τρεις γραμμές, με τις τιμές των μεταβλητών x, y, z μέσα και έξω από την παράλληλη περιοχή. Δώστε την έξοδο του κώδικα.*

```
int x = 1, y = 1, z = 1;
#pragma omp parallel shared(x) private(y) firstprivate(z) num_threads(2)
{
    x = 4;
    y = x + 1;
    z = x;
    printf("TID %d: x=%d y=%d z=%d\n", omp_get_thread_num(), x, y, z);
}
printf("x=%d y=%d z=%d\n", x, y, z);
```

TID 0: x=4 y=5 z=4
TID 1: x=4 y=5 z=4
x=4 y=1 z=1

Θέμα 13 (7μ: 1-4-2): *Δίνεται ο παρακάτω κώδικας σε CUDA:*

```
__device__ void mystery_kernel(int n, int *x)
{
    int i;
    for( i = 0; i < n; i++)
        x[i] = x[i] + 42;
}
```

που εκτελείται σε GPU από την main με την εντολή

```
mystery_kernel<<1,1>>(n, x);
```

οπού το x είναι διάνυσμα στην μνήμη της GPU.

Α. Το παραπάνω πρόγραμμα θα κάνει compile? Αν όχι, πού βρίσκεται το λάθος?

Β. Επισημάνετε ένα σημαντικό πρόβλημα στην επίδοσή του και διορθώστε το. Σας δίνεται η βοηθητική συνάρτηση:

```
__device__ int get_global_tid_1D(){ return threadIdx.x + blockDim.x * blockIdx.x;}
```

Γ. Είναι το συγκεκριμένο πρόβλημα ιδανικο για GPUs? Αιτιολογήστε την απάντηση σας.

Α. ΟΧΙ
- κώδικας __device__ δεν επιτρέπεται να καλείται από host(main)
- <<1, 1>> θα έπρεπε να είναι <<<1, >>>.
- <<1, 1>> δεν μειώνεται, πρέπει να ορίσουμε dim/dim3 μεταβλητές.

Β. Κάθε thread κάνει την ίδια δουλειά πάνω σε ένα τον πίνακα x και βάσει στην συγκεκριμένη περίπτωση κάνουμε allocate μόνο 1 thread.

```
mystery_kernel:
    int tid = get_global_tid_1D();
    if (tid >= n) return;
    x[tid] = x[tid] + 42;

main:
    dim3 b(32);
    dim3 g((n+31)/32);
    mystery_kernel<<<g, b>>>(n, x);
```

Γ. Όχι! Έχει πολύ μικρό O.I. για κάθε στοιχείο του πίνακα κάνουμε μόνο μία πρόσθεση.

Θέμα 14 (4μ): Επιλέξτε για κάθε έναν χαρακτηρισμό της αριστερής στήλης αν περιγράφει καλύτερα μία **CPU** ή μία **GPU.**

Υψηλή πολυπλοκότητα αρχιτεκτονικής για όλα τα προβλήματα.	CPU
Απλούστερη αρχιτεκτονική για ειδικά προβλήματα.	GPU
Πολλοί και αδύναμοι πυρήνες.	GPU
Λίγοι και ισχυροί πυρήνες.	CPU
'Γρήγορη' αλλά μικρότερη μνήμη.	GPU
'Αργή' αλλά μεγάλη μνήμη.	CPU
Λίγα νήματα ελεγχόμενα από το λειτουργικό.	CPU

Πολλά νήματα ελεγχόμενα από το hardware.	GPU

Θέμα 15 (10μ: 5 + 5): *Έστω ότι η Εφαρμογή #1 κυριαρχείται από την εκτέλεση του πολλαπλασιασμού πινάκων και η Εφαρμογή #2 από την εκτέλεση του πολλαπλασιασμού πίνακα με διάνυσμα που σας δίνονται στη συνέχεια (οι πίνακες περιλαμβάνουν δεδομένα των 4bytes).*

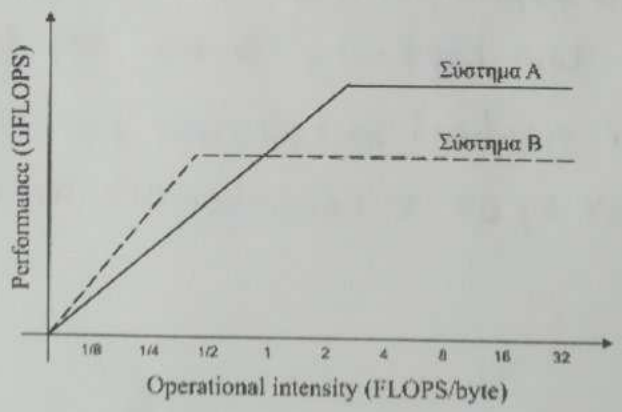
```
/* matrix multiplication */
for (i = 0; i < 100; i++)
   for (j = 0; j < 100; j++)
      for (k = 0; k < 100; k++)
      C[i][j] = C[i][j] + A[i][k] * B[k][j];

/* matrix-vector multiplication */
for (i = 0 ; i < 1000; i++)
   for (j = 0 ; j < 1000; j++)
      y[i] = y[i] + A[i][j] * x[j]
```

Α. Πόσους υπολογισμούς ανά δεδομένο (FLOPS/byte - operational intensity) κάνει κάθε εφαρμογή?

#1: Συνολικά $2 \cdot 10^6$ FLOPs (πολλαπλασιασμοί + προσθέσεις)
$2 \cdot 10^4 \cdot 4$ bytes δεδομένων (πίνακες A, B).
$\Rightarrow OI_1 = 100/4 = 25$

#2: $2 \cdot 10^6$ FLOPs (πολ/μού + προσθέσεις)
$10^6 \cdot 4 + 1000 \cdot 4$ bytes (πίνακας A, διάνυσμα x)
$\Rightarrow OI_2 \cong 1/4$

Β. Σας δίνεται επίσης το διάγραμμα roofline για δύο υποψήφια συστήματα Α και Β για την εκτέλεση των εφαρμογών. Επιλέξτε ένα σύστημα για κάθε εφαρμογή και αιτιολογήστε γιατί.



#1 → Α γιατί είναι computation heavy και το Α έχει μεγαλύτερο max performance.

#2 → Β είναι memory heavy (μικρό OI) ώστε μεγαλύτερη σημασία έχει η μνήμη, που έχει μεγαλύτερο bandwidth στο Β (κρίση σωθείσα).

Θέμα 16 (3μ): *Μια τεχνική παραλληλοποίησης βαθιών νευρωνικών δικτύων σε συστήματα κατανεμημένης μνήμης είναι ο παραλληλισμός pipeline. Δώστε ένα θετικό και ένα αρνητικό αυτής της τεχνικής, σε σχέση με το διαθέσιμο παραλληλισμό και το κόστος της επικοινωνίας.*

Θέμα 17 (6μ): *Η παραλληλοποίηση σε επίπεδο δεδομένων (data parallelism) σε βαθιά νευρωνικά δίκτυα σε κατανεμημένα συστήματα μπορεί να υλοποιηθεί με δύο τρόπους: με έναν κεντρικό εξυπηρετητή που διατηρεί όλες τις παραμέτρους (κεντρικός parameter server) ή με κατανεμημένους εξυπηρετητές των παραμέτρων (distributed parameter servers). Στην πρώτη περίπτωση, ένας μόνο εργάτης έχει επικαιροποιημένη εικόνα όλων των παραμέτρων σε κάθε βήμα, ενώ στη δεύτερη περίπτωση, όλοι οι εργάτες έχουν επικαιροποιημένη εικόνα των παραμέτρων σε κάθε βήμα. Δώστε ένα πλεονέκτημα και ένα μειονέκτημα της κάθε μεθόδου σε σχέση με την επικοινωνία.*

Κεντρικός parameter server

+ Ό,τι έχει ο server είναι το "ground truth" ⇒ δεν απαιτείται επικοινωνία για συγχρονισμό / consensus.

− Όλα πηγαίνουν από τον server ⇒ bottleneck επικοινωνίας, ⇒ δυσκολία στην κλιμάκωση

Distributed parameter servers

+ Πιο αποκεντρωμένο, το "βάρος" της επικοινωνίας μοιράζεται ισότιμα σε όλους τους κόμβους / servers, άρα καλύτερη κλιμάκωση.

− Επιπρόσθετο overhead επικοινωνίας για τη διασφάλιση ότι όλοι οι εργάτες έχουν επικαιροποιημένη εικόνα των παραμέτρων.

[redacted]

ΕΜΠ - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μετ.. Πρόγραμμα «Επιστήμη Δεδομένων και Μηχανική Μάθηση»

ΑΘΗΝΑ 16 Ιουνίου 2023

Μάθημα: Παράλληλες Αρχιτεκτονικές Υπολογισμού για Μηχανική Μάθηση

[redacted]

Θεωρια MICROLAB: 2023

(α) Να περιγράψετε τα κύρια μέρη μιας αρχιτεκτονικής FPGA. (Σημείωση: Ένα σχήμα θα ήταν πολυ επιθυμητό) *(4 μονάδες)*

(β) Ποια η διαφορά της BlockRAM (BRAM) μνήμης προγραμματισμού SRAM; *(2 μονάδες)*

(γ) Ποιός ειναι ο ρόλος του bitstream σε ένα FPGA; *(2 μονάδες)*

(δ) Ποιός ειναι ο ρόλος των DSP/multiplication μονάδων; *(2 μονάδες)*

(ε) Να δοθεί ο ορισμός του επιταγχυντή υλικού (accelerator); Ποιά/ιές ανάγκη/ες επέβαλε/αν την χρήση τους; *(5 μονάδες)*

(στ) Να περιγράψετε την αρχιτεκτονική ενός SoC-FPGA. Να εξηγησετε την εννοια του συ-σχεδιασμού υλικού και λογισμικού (Hw/Sw Codesign). *(5 μονάδες)*

ΑΣΚΗΣΕΙΣ MICROLAB: 2023

Θεμα 1 (10 μονάδες):

Μια εφαρμογή μηχανικής μάθησης αποτελείται από 3 συναρτήσεις f1, f2 και f3. Σε ένα τυπικό επεξεργαστή γενικού σκοπού (CPU) ο χρόνος εκτέλεσης της συνάρτησης f1 είναι 10 ώρες, ο χρόνος εκτέλεσης της συνάρτησης f2 είναι 40 ώρες και ο χρόνος εκτέλεσης της τρίτης συνάρτησης f3 είναι 50 ώρες.

Θέλετε να αυξησετε την ταχύτητα εκτέλεσης της εφαρμογής και έχετε 2 επιλογές.

Α) Να χρησιμοποιήσετε μια GPU με 6 πυρήνες για την συνάρτηση f2 (πληρως παραλλελοποιήσιμη) και οι υπόλοιπες συναρτήσεις να τρέξουν στην CPU

Β) Να χρησιμοποιήσετε μια FPGA η οποία μπορεί να εκτελέσει την συνάρτηση f3 3x φορές πιο γρήγορα από ότι ο τυπικός επεξεργαστής και οι υπολοιπες συναρτήσεις να τρέξουν στην CPU.

Να υπολογίσετε το Speedup σε κάθε περίπτωση και να βρείτε ποια επιλογή είναι καλύτερη.

Θεμα 2 (10 μονάδες):

Χρειάζεται να χρησιμοποιήσετε ένα νευρωνικο δικτυο για την επεξεργασία 20.000 εικονων (inference). Εχετε τις παρακάτω επιλογες ως υπολογιστική πλατφόρμα.

- CPU: 2000 images/ωρα, $3/ωρα, 100W
- GPU: 4000 images/ωρα, $5/ωρα, 200W
- FPGA: 5000 images/ωρα, $6/ωρα, 300W

Α) Ποια πλατφόρμα από αυτές θα είναι πιο οικονομική για την συγκεκριμένη εφαρμογή;

Β) Υπολογίστε την ενέργεια που θα καταναλώσει κάθε πλατφόρμα σε kWh.