05-02-2024

Παράλληλος προγραμματισμός

C code vs Python vs Numba

Εισαγωγή

Σύμφωνα με τον "Νόμο του Μουρ" που διατυπώθηκε για πρώτη φορά το 1965 ο αριθμός των τρανζίστορ σε ένα ολοκληρωμένο κύκλωμα θα διπλασιάζεται κάθε δυο χρονιά, και όπως διαπιστώνουμε από το παρακάτω διάγραμμα (Figure 1) αυτό ισχύει έως και σήμερα, ίσως με μια ελαφριά μείωση του ρυθμού. Αυτό το γεγονός είναι πολύ θετικό γιατί η εκθετική αυτή αύξηση των τρανζίστορ συνεπάγεται εκθετική αύξηση της επίδοσης των υπολογιστών και γενικότερα της τεχνολογίας. Βέβαια, το αρνητικό σε όλο αυτό είναι ότι εκθετικά αυξάνεται και η κατανάλωση ενέργειας που απαιτείται για να λειτουργήσουν αυτά τα συστήματα. Αυτή η προσέγγιση εξέλιξης της απόδοσης των υπολογιστών έγινε γρήγορα αντιληπτή πως δεν είναι βιώσιμη. Για να αντιμετωπιστεί αυτό το πρόβλημα άρχισαν οι επεξεργαστές να έχουν πόλους πυρήνες ώστε να αυξάνεται η συνολική απόδοση χωρίς να υπάρχει μεγάλη αύξηση στην απαιτούμενη ισχύεις. Το πρόβλημα με αυτό είναι πως η εκμετάλλευση των πολλών πυρήνων απαιτεί εξειδικευμένο προγραμματισμό των λογισμικών καθώς και δεν μπορούν όλα τα λογισμικά εκ φυσικού τους να τους εκμεταλλευτούν. Αυτό το γεγονός έχει ως αποτέλεσμα τα άτομα που έχουν τους υπολογιστές τους για απλή χρήση τα οποία είναι και η πλειοψηφία, να μην χρησιμοποιούν πέρα από ένα μικρό ποσοστό της επεξεργαστικής του ισχυρής παρότι μπορεί να έχουν αγοράσει ένα πολύ ακριβό και ισχυρό σύστημα. Στους επιστημονικούς υπολογισμούς όμως, που είναι και η κύρια χρήση της τόσο μεγάλης και συνεχώς αυξανόμενης υπολογιστικής ισχύεις αναπτύσσονται συνέχεια μέθοδοί εκμετάλλευσης της, ώστε να μειώνονται οι χρόνοι στους υπολογισμούς καθώς και στην αύξηση της ακρίβειας των αποτελεσμάτων.

Σε αυτή την εργασία θα κάνουμε χρήση της υπολογιστικής μεθόδου Monte Carlo για τον υπολογισμό της μαθηματικής σταθεράς **π**. Η υλοποίηση θα γίνει σε γλώσσα προγραμματισμού **c**, η οποία θεωρείται μια από τις πιο γρήγορες γλώσσες για επιστημονικούς υπολογισμούς. Στην υλοποίηση αυτή θα γίνει χρήση του λογισμικού OpenMP για παραλληλοποίηση του προγράμματος προς μείωση του χρόνου εκτέλεσης. Τέλος τα αποτελέσματα θα συγκριθούν με μια υλοποίηση της μεθόδου Monte Carlo σε Python η οποία θεωρείται μια πολύ αργή γλώσσα προγραμματισμού, και πως αυτή βελτιώνεται με την βιβλιοθήκη Numba η οποία κάνει χρήση παράλληλου προγραμματισμού,

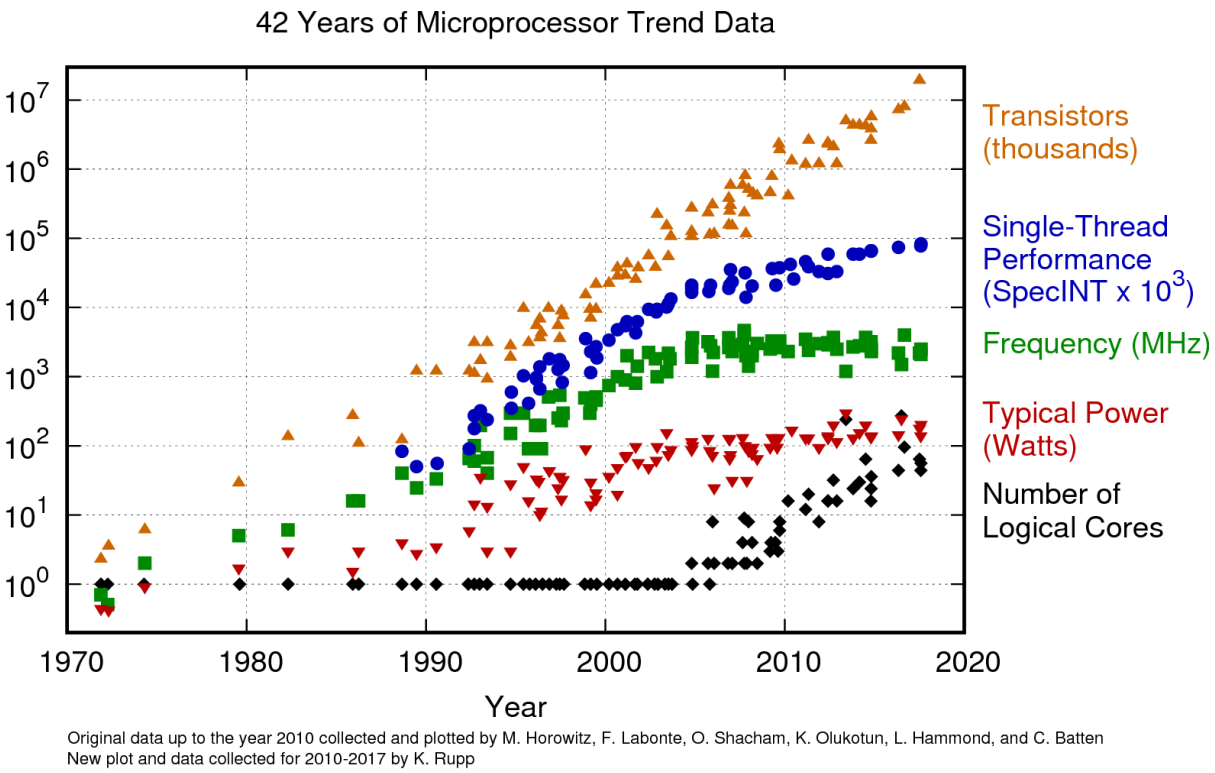

Figure 1 *Στο διάγραμμα αυτό φαίνεται η αύξηση του αριθμού των τρανζίστορ, η απόδοση ενός πύρινα, η συχνότητα ενός πύρινα, η ισχύς που απαιτείται, καθώς και οι αύξηση τον λογικών πύρινων ανά τα χρονιά.*

Monte Carlo

Η ονομασία μέθοδος Monte Carlo ή προσομοίωση Monte Carlo, δεν είναι μια συγκεκριμένη μέθοδος αλλά μια γενική ιδέα προσέγγισης διάφορων προβλημάτων, που κείνο έχουν την χρήση τυχαίων αριθμόν. Στην περίπτωση μας για τον υπολογισμό του π, θα χρησιμοποιήσουμε την προσομοίωση Monte Carlo που υπολογίζει ολοκληρώματα. Η βασική ιδέα για τον υπολογισμό ολοκληρωμάτων με την μέθοδο αυτή, είναι η παραγωγή τυχαίων αριθμών από ομοιόμορφη κατανομή σε ένα πλαίσιο μεγαλύτερο από το εμβαδό ολοκλήρωσης που μας ενδιαφέρει (Figure 2). Συνηθώς το πλαίσιο αυτό είναι ορθογώνιο παραλληλόγραμμο γιατί η παραγωγή τυχαίων αριθμών για τα σημεία της επιφάνειας του είναι πολύ απλή. Χρησιμοποιώντας δυο αριθμούς, έναν που να αναφέρεται στον x άξονα και έναν για τον y άξονα, μπορούμε πολύ απλά να αναπαραστήσουμε την επιφάνια του ορθογώνιού παραλληλογράμμου. Παράγοντας έναν πολύ μεγάλο αριθμό ζευγαριών σημείων x,y θα καταλήξουμε σε μια εικόνα όπως αυτή στο Figure 2 αλλά με πολλά περισσότερα σημεία. Η αναλογία των σημείων που είναι μέσα στο προς ολοκλήρωση εμβαδό (κόκκινα) προς το σύνολο, είναι ίση με την αναλογία στο εμβαδό του προς ολοκλήρωση σχήματος προς του ορθογωνίου (1).

$$\frac{\text{αριθμός σημείων μέσα}}{\text{σύνολο σημείων}} = \frac{I}{I_{ορθ.}} \Rightarrow$$

$$I = \frac{N_{μέσα}}{N} I_{ορθ.} \qquad (1)$$

Είναι γνωστό πως το εμβαδό ενός τετραγώνου με πλευρά $2r$ είναι $4r^2$, ενώ ενός εγγεγραμμένου κύκλου σε αυτό το τετράγωνο είναι πr^2. Από την εξίσωση 1 προκύπτει πως:

$$\pi r^2 = \frac{N_{μέσα}}{N} 4r^2 =>$$

$$\pi = 4\frac{N_{μέσα}}{N} \quad (2)$$

Με την εξίσωση 2 μπορούμε πολύ απλά παράγοντας τυχαίους αριθμούς σε ένα πλαίσιο και καταμετρώντας τα σημεία που πέφτουν μέσα σε αυτό το πλαίσιο να προσεγγίσουμε την τιμή του π.

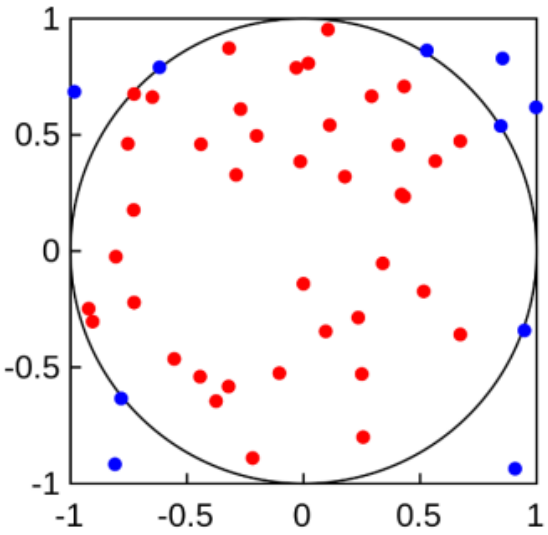

Figure 2 *Αναπαράσταση της χρήσης της τεχνικής ολοκλήρωσης Μόντε Κάρλο για τον υπολογισμό του εμβαδού ενός κύκλου εγγεγραμμένου σε ένα τετράγωνο.*

Αλγοριθμικά βήματα

1. Επανάληψη βημάτων 2 έως 4, N φορές.
2. Παραγωγή τυχαίου αριθμού από ομοιόμορφη κατανομή στο διάστημα [0, 1] και ορισμός τους ως rx.
3. Παραγωγή τυχαίου αριθμού από ομοιόμορφη κατανομή στο διάστημα [0, 1] και ορισμός τους ως ry.
4. Έλεγχος αν $rx^2 + ry^2 \leq 1$.
   4.1. Αν είναι αληθές, $N_{in} = N_{in} + 1$.
5. Εκτύπωση, $4\,N_{in}/N$.

Αποτελέσματα

Τα αποτελέσματα έχουν υπολογιστεί σε σύστημα με 6 φυσικούς πύρινες και 2 threads ανά πυρήνα. Στο Figure 3, μπορούμε να παρατηρήσουμε την μείωση τους χρόνου εκτέλεσης του κώδικα. Η μείωση αυτή δεν είναι γραμμική και μόνο οι φυσικοί πυρήνες συνεισφέρουν ουσιαστικά, ενώ αντίθετα, οι λογικοί πυρήνες συνεισφέρουν από ελάχιστα έως και καθόλου σε αυτό. Αυτό μπορούμε να το παρατηρήσουμε πολύ έντονα στην καμπύλη $N = 10^9$ όπου μετρά τα 6 πρώτα threads που αντιστοιχούν στους φυσικούς πυρήνες, η συνέχεια της καμπύλης είναι σχεδόν οριζόντια και τα όποια μικρά σκαμπανεβάσματα είναι στα πλαίσια του σφάλματος. Την ίδια ακριβός συμπεριφορά θα παρατηρούσαμε και από τις υπόλοιπες καμπύλες εάν άλλαζε η κλίμακα.

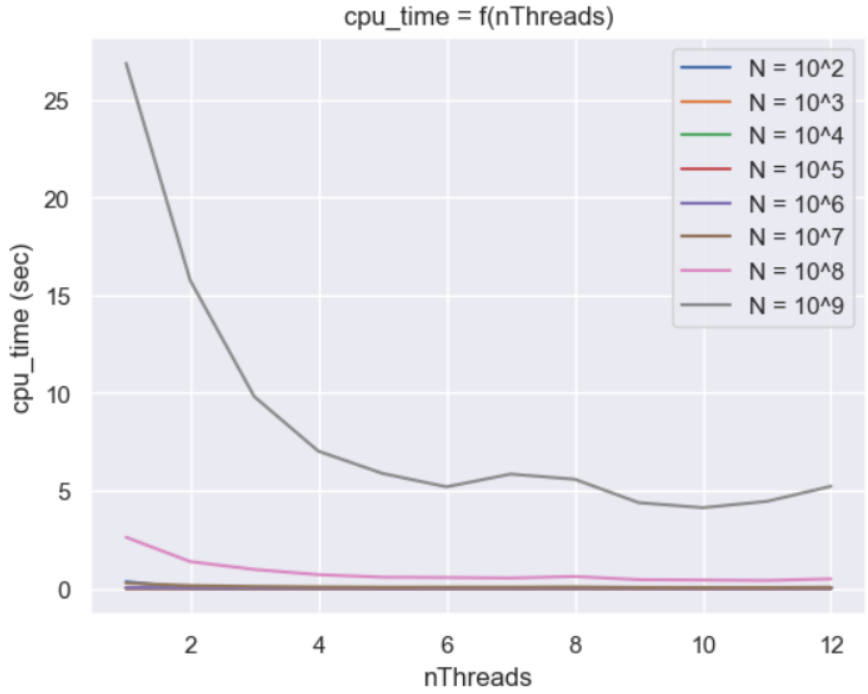

Figure 3 *Μείωση του χρόνου εκτέλεσης κώδικα ως συνάρτηση με τα threads (σύστημα με 6 φυσικούς πύρινες).*

Στο Figure 4, για $N \geq 10^7$ παρατηρούμε ότι αύξηση του χρόνου εκτέλεσης του κώδικα φαίνεται αν είναι γραμμικά εξαρτημένος από τον αριθμό των σημείων, αλλά πρέπει να προσέξουμε ότι οι άξονες δεν αυξάνουν γραμμικά αλλά εκθετικά, συνεπώς η συσχέτιση του χρόνου εκτέλεσης και τους αριθμού των σημείων, ακολουθούν νομό δύναμης (3). Ο λόγος που για μικρότερες τιμές αυτή η σχέση φαίνεται να μην ισχύει είναι απλός επειδή ο χρόνος εκτέλεσης είναι τόσο μικρός που τα σφάλματα υπερισχύουν κατά πολύ σε αυτήν την περιοχή.

$$y = Ax \Rightarrow 10^y = 10^{Ax} \Rightarrow 10^y = (10^x)^A$$

$$cpu\ time = N^A \tag{3}$$

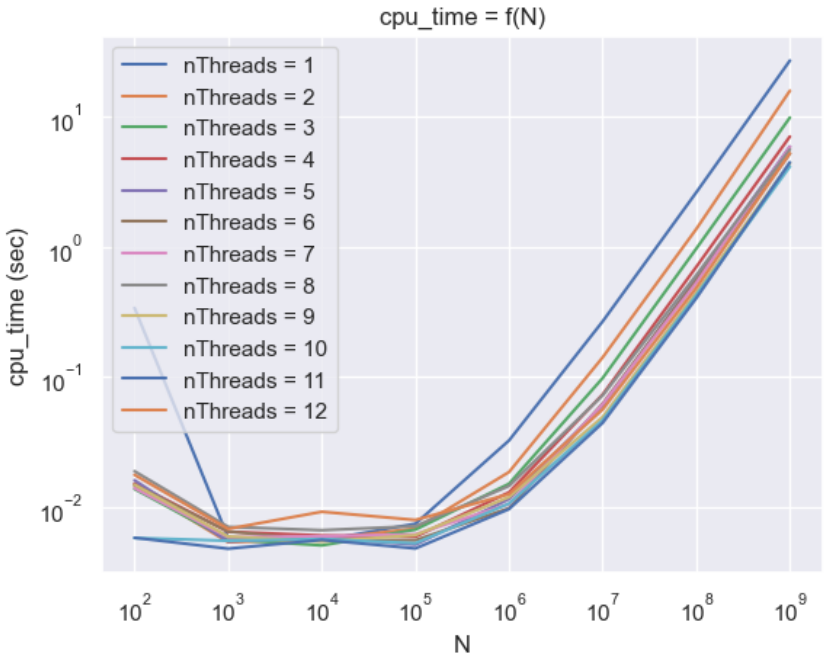

Figure 4 *Χρόνος εκτέλεσης κώδικα σε συνάρτηση με τον αριθμό επαναλήψεων (σύστημα με 6 φυσικούς πύρινες).*

Στο Figure 5 βλέπουμε πως μεταβάλλεται η επιτάχυνση του κώδικα όσο αυξάνουν οι φυσικοί πυρήνες. Τα σημεία έχουν υπολογιστεί ως μέσες τιμές της επιτάχυνσης της κάθε καμπύλης με $N \geq 10^7$ για κάθε πυρήνα. Αυτό έγινε για να μειωθούν τα σφάλματα των αποτελεσμάτων. Η καμπύλη είναι η θεωρητική πρόβλεψη από το νομό του Amdahl, η οποία για τα δεδομένα μας προκύπτει ότι το ποσοστό του κώδικα που επωφελείται από την παραλληλοποίηση είναι περίπου 95.8%, ενώ προβλέπει πως για 1000 φυσικούς πυρήνες η επιτάχυνση θα ήταν περίπου 23.9 φορές μεγαλύτερη από τον 1 πυρήνα.

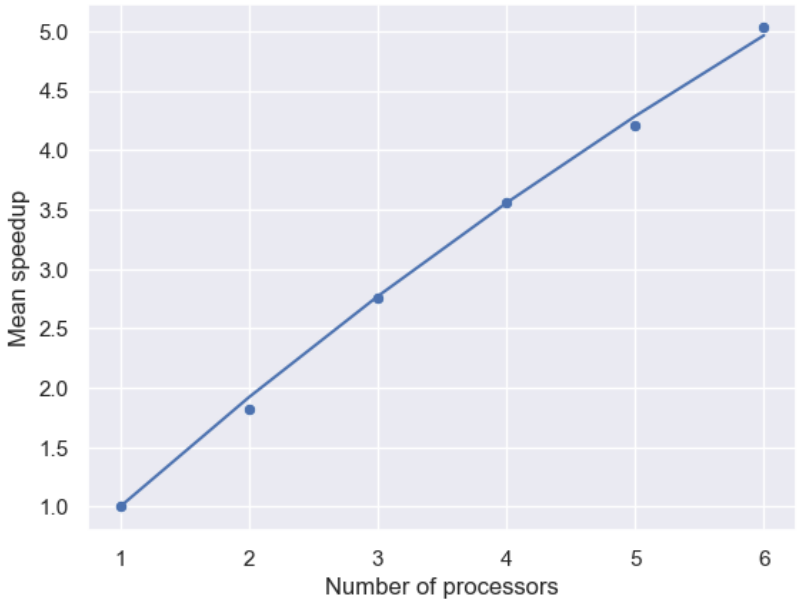

Figure 5 *Επιτάχυνση του κώδικα με την αύξηση των φυσικών πυρήνων.*

Στο Figure 6 έχουμε την μείωση του απόλυτου σφάλματος από την πραγματική τιμή του π. Η ευθεία (πορτοκαλή γραμμή) μας δείχνει την θεωρητική μείωση του απόλυτου σφάλματος. Να σημειωθεί το γεγονός πως, οι τιμές του σφάλματος φαίνονται να απομακρύνονται περισσότερο προς τα κάτω από την θεωρητική ευθεία, αλλά αυτό είναι παραπλανητικό, καθώς οι άξονες αυξάνουν εκθετικά. Βάση αυτού προκύπτει και πάλι πως η σχέση του απόλυτου σφάλματος με τον αριθμό των σημείων μεταβάλλονται με νόμο δύναμης, και αφού η κλήση είναι αρνητική ο εκθέτης πρέπει να είναι αρνητικός, συνεπώς επιβεβαιώνεται μέχρι ενός σημείου η θεωρητική σχέση η οποία έχει εκθέτη -0.5. Τα αποτελέσματα θα βελτιωνόντουσαν κατά πολύ, αν εφαρμόζαμε άλλη μια τεχνική Monte Carlo οπού προσομοιώνουμε το ίδιο ακριβός φαινόμενο πολλές φορές, και έπειτα περνούμε τις μέσες τιμές και το σφάλμα του δείγματος. Με αυτό τον τρόπο θα μειωνόντουσαν οι έντονες διακυμάνσεις γύρο από τις θεωρητικές τιμές, επειδή θα μειωνόντουσαν τα σφάλματα. Στο Figure 7 βλέπουμε την συμπληρωματική εικόνα του Figure 6, δηλαδή, το πως η τιμή της προσομοιώσεις προσεγγίζει την πραγματική τιμή του π.

Figure 6 *Διάγραμμα του απόλυτου πραγματικού σφάλματος της προσομοίωσης για διαφορετικό αριθμό σημείων.*

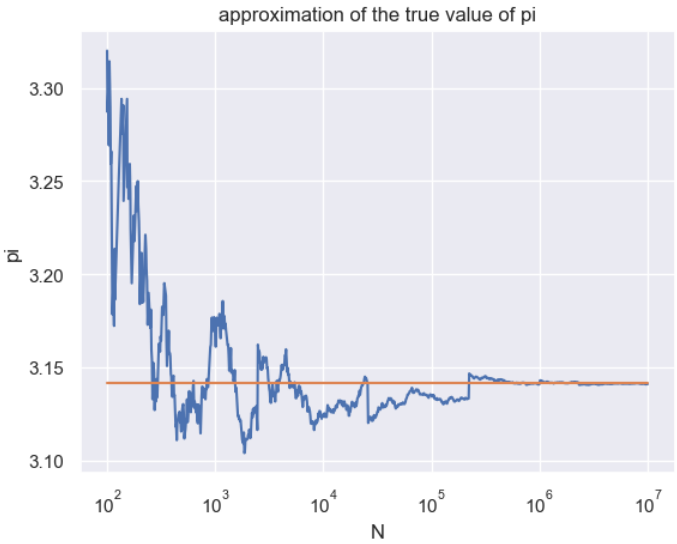

Figure 7 *Προσέγγιση της πραγματικής τιμής του π με την αύξηση των τυχαίων σημείων.*

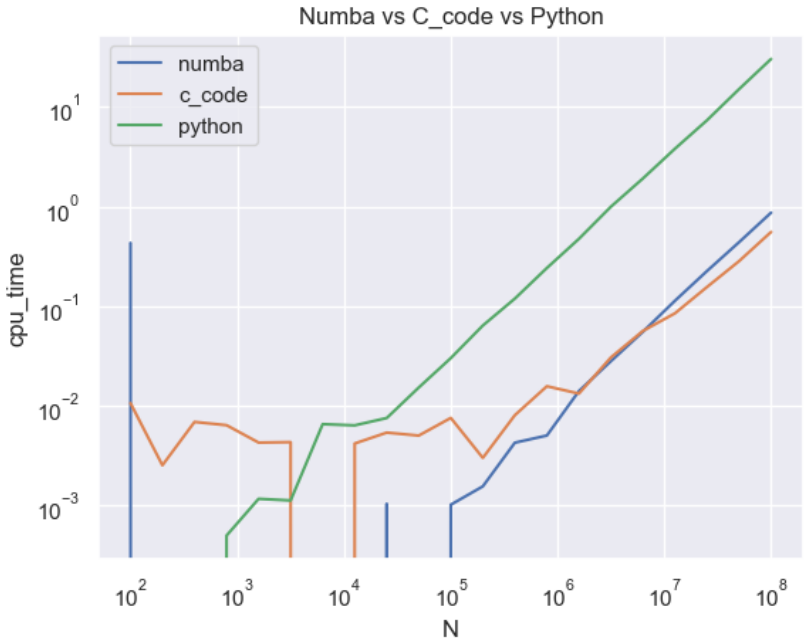

Figure 8 Χρόνος εκτέλεσης κώδικα «παράλληλα» με τρις διαφορετικούς τρόπους, σε Python, σε Python με την βιβλιοθήκη Numba και σε C/C++.

Τέλος, έχουμε την σύγκριση στην επίδοση της γλώσσα Python, της Python με την βιβλιοθήκη Numba και της γλώσσας C/C++. Να σημειωθεί ότι επειδή στην απλή Python (χωρίς την βιβλιοθήκη Numba) δεν έχει εφαρμοστεί παράλληλος προγραμματισμός, έχουμε διαιρέσει τους χρόνους εκτελέσεις για κάθε N με τον αριθμό των φυσικών πυρήνων, και αυτό, για να έχουμε μια πιο ορθή σύγκριση των αποτελεσμάτων, παρόλο που γνωρίζουμε πως ο χρόνος εκτέλεσης δεν μειώνεται γραμμικά με τους πυρήνες όπως είδαμε και παραπάνω (Figure 5), παρόλα αυτά είναι μια πρώτης τάξης προσέγγιση. Από το διάγραμμα (Figure 6) παρατηρούμε πως για μικρα N, η Python και η Numba υπερισχύουν. Αυτό οφείλεται στο ότι για μικρές τιμές του N οι χρόνοι εκτέλεσης είναι αρκετά μικροί και συνεπώς τα σφάλματα να είναι έντονα. Επιπρόσθετα σε αυτό έρχεται να συμβάλει και το γεγονός πως σε κάθε προσομοίωση στην C/C++, έχουμε έναν επιπρόσθετο μικρό χρόνο που στην Python δεν υπάρχει που είναι ο χρόνος ορισμού τον μεταβλητών. Για μεγάλες τιμές του N που τα σφάλματα και οι λοιποί χρόνοι, δεν συμβάλουν ουσιαστικά και οι καμπύλες φαίνονται να κινούνται ευθεία, έχουμε τα εξής αποτελέσματα βάση των δεδομένων μας:

- Η Numba είναι 483.8% πιο γρήγορη από την Python.
- Η C/C++ είναι 812.0% πιο γρήγορη από την Python.
- Η C/C++ είναι 52.2% πιο γρήγορη από την Numba.

Συμπεράσματα

Αυτό που μπορούμε να συμπεράνουμε από τα αποτελέσματα είναι σαφές, και είναι το ότι ο παράλληλος προγραμματισμός γενικά, και σε συνδυασμό με μια γλώσσα όπως η C/C++ ή την Fortran ιδικά, μπορεί να μας γλυτώσει πάρα πολύ πολύτιμο χρόνο τον οποίον μπορούμε να τον αφιερώσουμε κάπου αλλού. Από την άλλη όμως, όπως είδαμε στο Figure 5 βάση του νομού το Amdahl, δεν μπορούμε να επιταχύνουμε τον κώδικα μας πέρα από μια τιμή, και η χρήση πολύ μεγάλου αριθμού πυρήνων είναι σκέτη σπάταλη υπολογιστικών πόρων και κατά συνέπεια ενεργείας, αφού χρειάζονται πολύ πυρήνες για μια πολύ μικρή μείωση στον χρόνο εκτέλεσης, συνεπώς ασύμφορο.