Hybrid K-means with OpenMP and MPI

Παράλληλος και Δικτυακός υπολογισμός

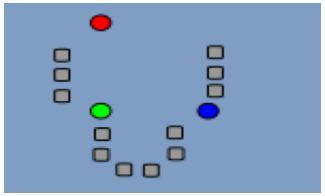
Ασημίνα Βουρονίκου - Ελένη Ζησιού

Γενικά ο αλγόριθμος(1/2)

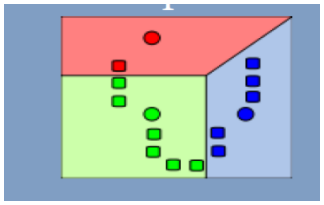
Ο αλγόριθμος αποτελείται από τρία βασικά βήματα:

1. Τοποθέτησε Κ σημεία στο ίδιο χώρο όπου είναι συγκεντρωμένα τα δεδομένα προς συσταδοποιηση. Τα σημεία αυτά αποτελούν αρχικά τα centroids των clusters
2. Αντιστοίχισε κάθε δεδομένο στο cluster που έχει το πιο κοντινό κέντρο βάρους. Όταν όλα τα δεδομένα έχουν ανατεθεί, υπολόγισε εκ νέου τις θέσεις των Κ - centroids
3. Επανέλαβε τα βήματα 2 και 3 μέχρι τα centroids πλέον να μην μετακινούνται. Αυτό παράγει ένα διαχωρισμό των αντικειμένων σε clusters από τις οποίες η μετρική που πρέπει να ελαχιστοποιηθεί μπορεί πλέον να υπολογιστεί

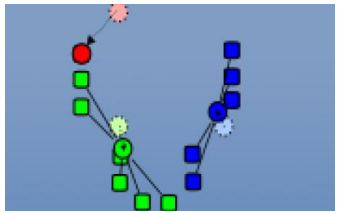
Γενικά ο αλγόριθμος(2/2)



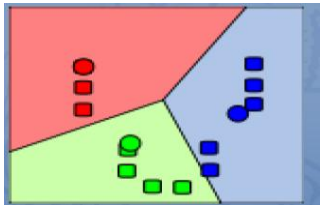
Κ -σημεία διαλέγονται τυχαία ως κέντρα



K-clusters δημιουργούνται συνδέοντας κάθε σημείο με το κοντινότερο κέντρο



Υπολογίζεται το νέο κέντρο του κάθε cluster



Επανάληψη μέχρι να μην υπάρχει μετακίνηση

Πολυπλοκότητα του αλγορίθμου

- Η πολυπλοκότητα είναι O(n*k*I*d)
  - n = αριθμός αντικειμένων
  - k = αριθμός clusters
  - I = αριθμός επαναλήψεων
  - d = διάσταση

- Για σταθερά k,d ο αλγόριθμος παρατηρείται ότι μπορεί να ολοκληρώσει το clustering σε ακριβώς **$O(n^{dk+1} \log n)$** χρόνο

Ανάγκη παραλληλισμού

- Τα αντικείμενα είναι ανεξάρτητα μεταξύ τους ⟶ εγγενώς παράλληλοποιήσιμος
- Είναι σημαντικά ευαίσθητος στα αρχικά τυχαία κέντρα που επιλέγονται
  - Χρειάζεται μια «πονηρή» τοποθέτηση
- Όσο το σύνολο δεδομένων κλιμακώνει γίνεται όλο και πιο δύσκολο να χρησιμοποιήσουμε τον k-means για ένα τόσο μεγάλο ποσό δεδομένων

- Μια **υβριδική λύση** προγραμματίζοντας σε **OpenMP και MPI**.
- MPI ⟶ επικοινωνία μεταξύ των clusters
- OpenMP ⟶ παραλληλοποίηση της δουλειάς στο εσωτερικό των κόμβων

Βελτιστοποιήσεις σειριακού κώδικα

- **Αρχικοποίηση των centroids**

  Ο αλγόριθμος αρχικοποίησης KKZ:

  1. Υπολόγισε τις νόρμες όλων των διανυσμάτων στο dataset. Διάλεξε το vector με την μεγαλύτερη νόρμα ως πρώτο centroid.
  2. Υπολόγισε τις αποστάσεις όλων των διανυσμάτων στο dataset από το πρώτο centroid. Διάλεξε το vector με την μεγαλύτερη απόσταση ως δεύτερο centroid.
  3. Για ένα σύνολο από centroids μεγέθους i με i=2,3 …υπολόγισε την απόσταση κάθε διανύσματος από κάθε centroid και κράτα την μικρότερη. Από όλες τις αποστάσεις η μεγαλύτερη είναι το (i+1) centroid. Συνέχισε μέχρι να υπολογιστούν K- centroids.

- **Συνθήκη τερματισμού του αλγορίθμου**
  - Γενικά οι επαναλήψεις σταματούν με βάση ένα tolerance
  - Στην υλοποίησή μας οι επαναλήψεις να σταματούν όταν δεν παρατηρείται καμία κίνηση
    - ✓ πιο αποδοτικός
    - ✓ καλύτερη δυνατή τοποθέτηση των αντικειμένων στα clusters

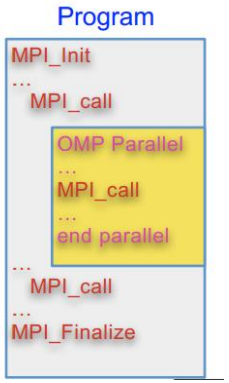
Παραλληλοποίηση με OpenMP

- **Στρατηγική παραλληλοποίησης**
  - Ο υπολογισμός του κοντινότερου cluster για κάθε αντικείμενο γίνεται παράλληλα με OpenMP parallel for
  - Η μέτρηση των αλλαγών στα clusters γίνεται ατομικά με omp atomic.
  - Η άθροιση των αντικειμένων του κάθε cluster καθώς και η μέτρηση των εμφανίσεων τους γίνεται ατομικά με omp atomic.
  - Ο υπολογισμός των νέων centroids γίνεται παράλληλα με OpenMP parallel for.

Υβριδική Υλοποίηση OpenMP-MPI

- **Στρατηγική παραλληλοποίησης**

  - Αρχικοποίηση του MPI μοντέλου
  - Διαμοιρασμός των αντικειμένων στα clusters
  - Σε κάθε επανάληψη στο εσωτερικό του κάθε κόμβου:
    - Ο υπολογισμός του κοντινότερου cluster για κάθε αντικείμενο γίνεται παράλληλα με OpenMP parallel for
    - Η μέτρηση των αλλαγών στα clusters γίνεται ατομικά με omp atomic.
    - Η άθροιση των αντικειμένων του κάθε cluster καθώς και η μέτρηση των εμφανίσεων τους γίνεται ατομικά με omp atomic



  - Στο τέλος κάθε επανάληψης έχουμε reduction των αποτελεσμάτων
  - Κόμβος 0: υπολογισμός των νέων centroids, broadcast στους υπολοίπους
  - Οι επαναλήψεις συνεχίζονται μέχρι να μην υπάρχει μετακίνηση στα clusters.

Μετρήσεις

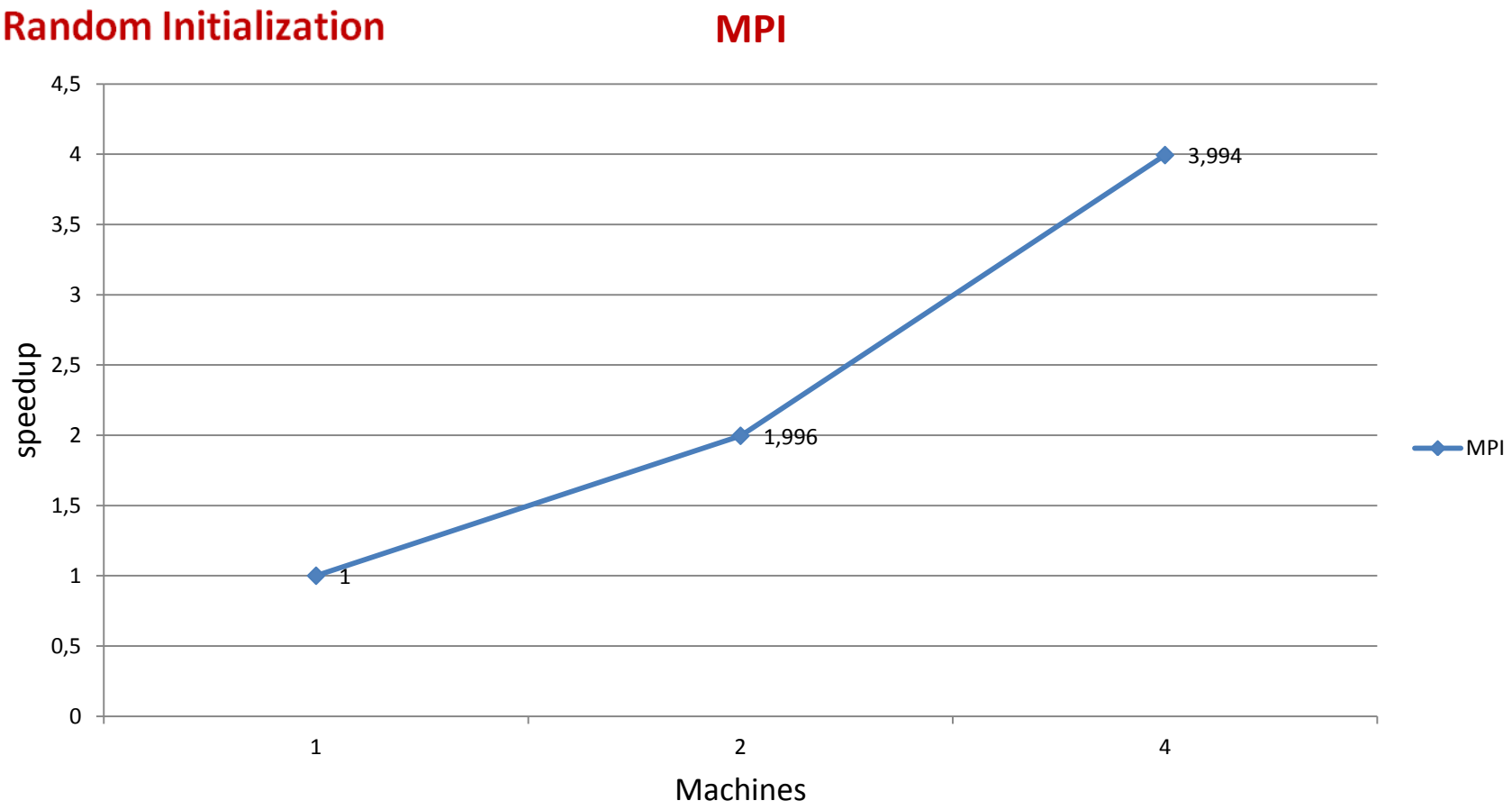
- Cluster 4 πολυπύρηνων μηχανημάτων.
  - Κάθε μηχάνημα διαθέτει 4 πυρήνες , 8 GB RAM ,Intel Xeon W3550 @3.07GHz επεξεργαστή.

- Οι μετρήσεις πραγματοποιήθηκαν για 4 τάξεις μεγέθους data set σε clusters 1,2 και 4 μηχανημάτων και για 1,2 και 4 threads ανά μηχάνημα.

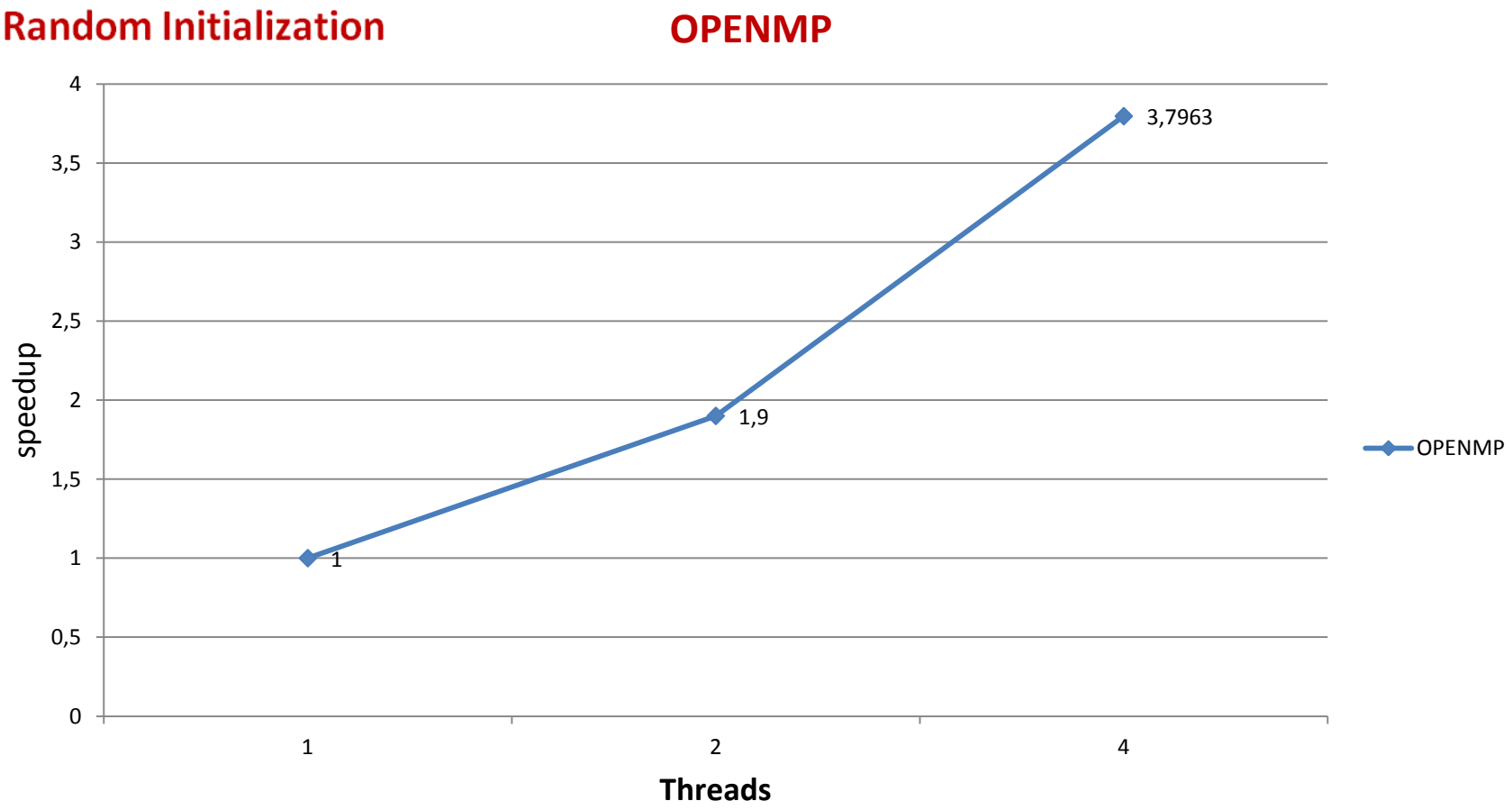
Συγκρίσεις :

- KKZ Initialization vs. Random Initialization
- Hybrid vs. OpenMP
- Hybrid vs. MPI
- Hybrid vs. Serial
- MPI vs. OpenMP

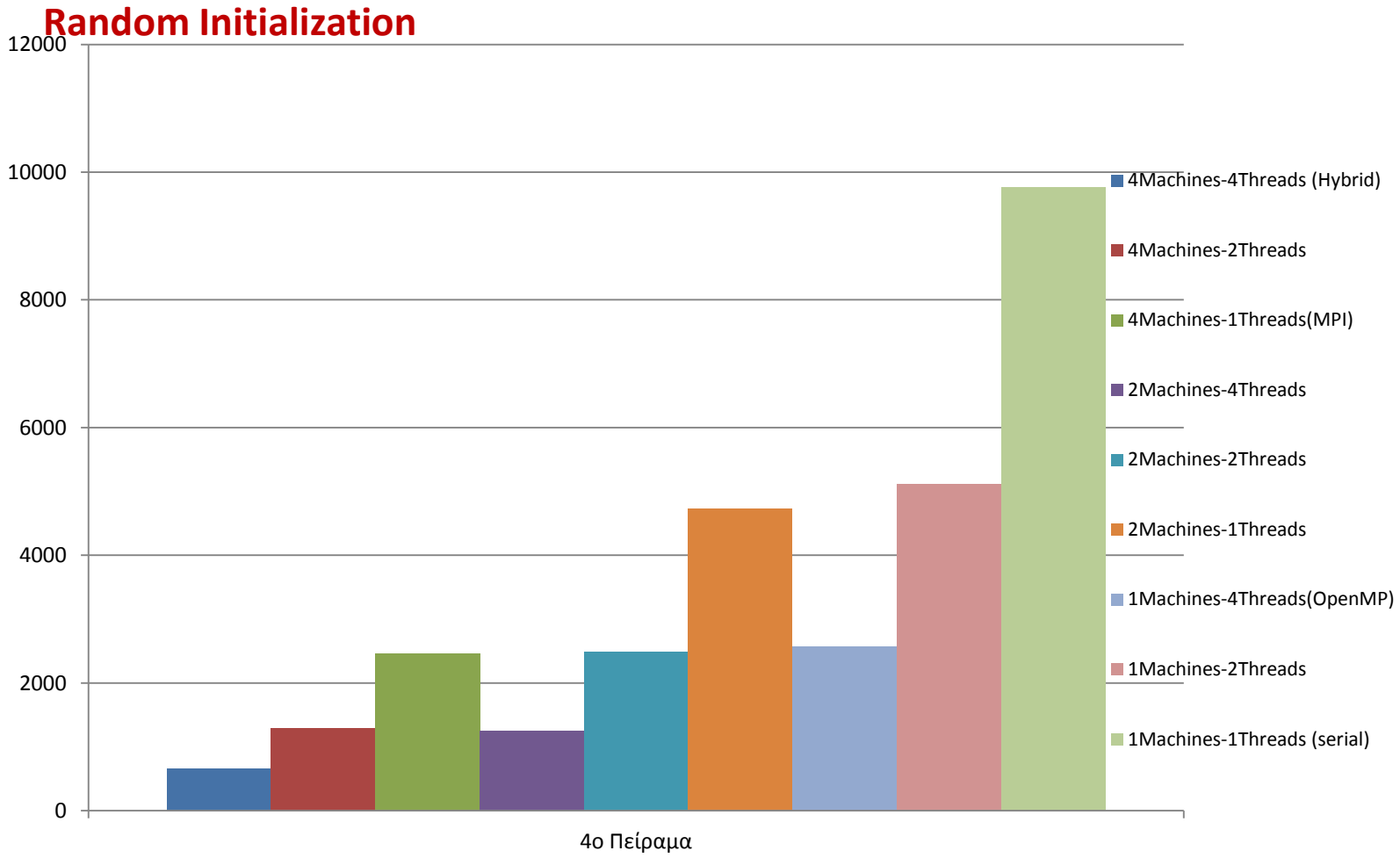
Pure MPI υλοποίηση- Speedup



Pure OpenMP υλοποίηση- Speedup

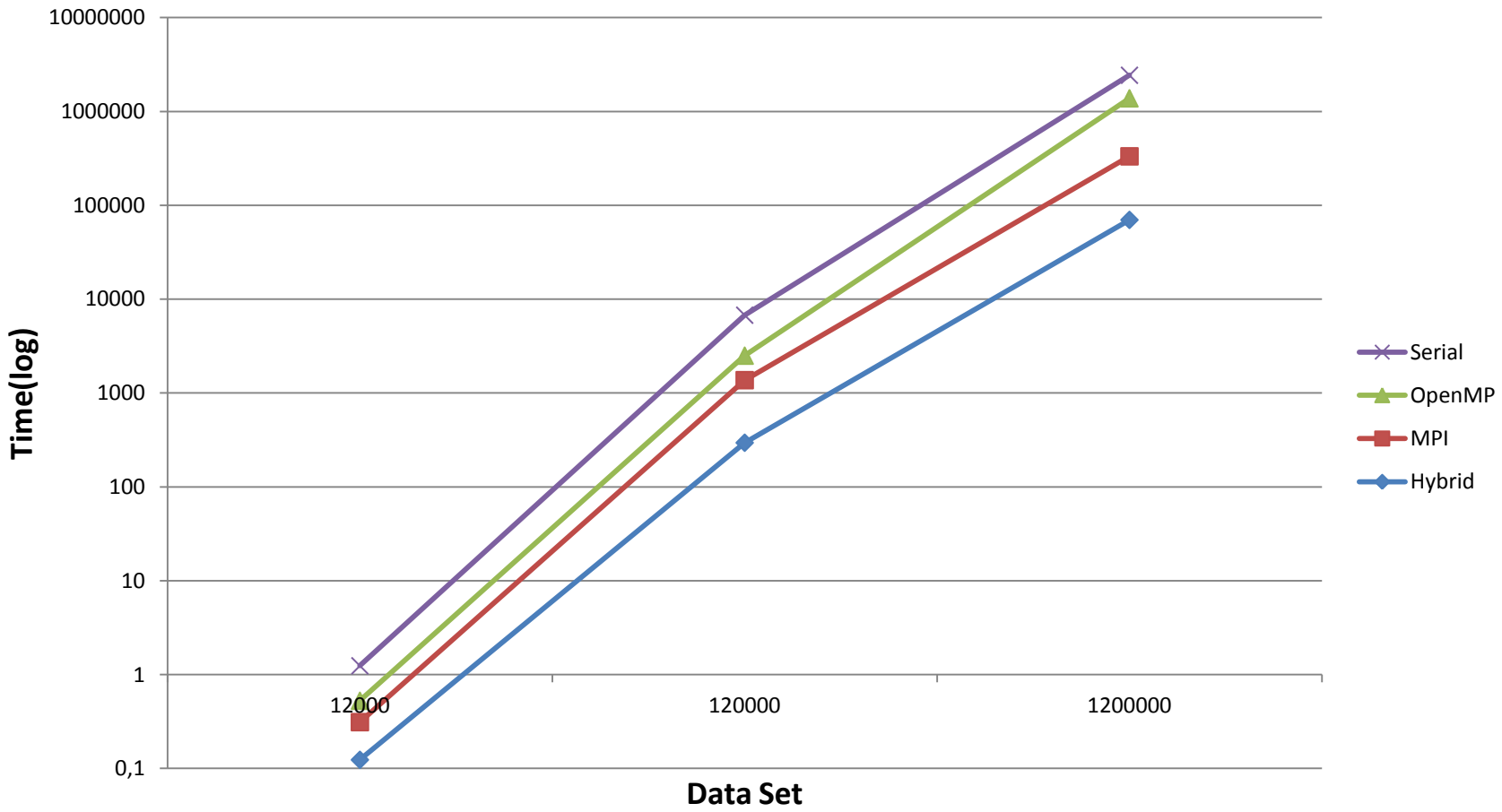


Αντικείμενα: 1.200.000, διάσταση: 16, κέντρα: 10.000



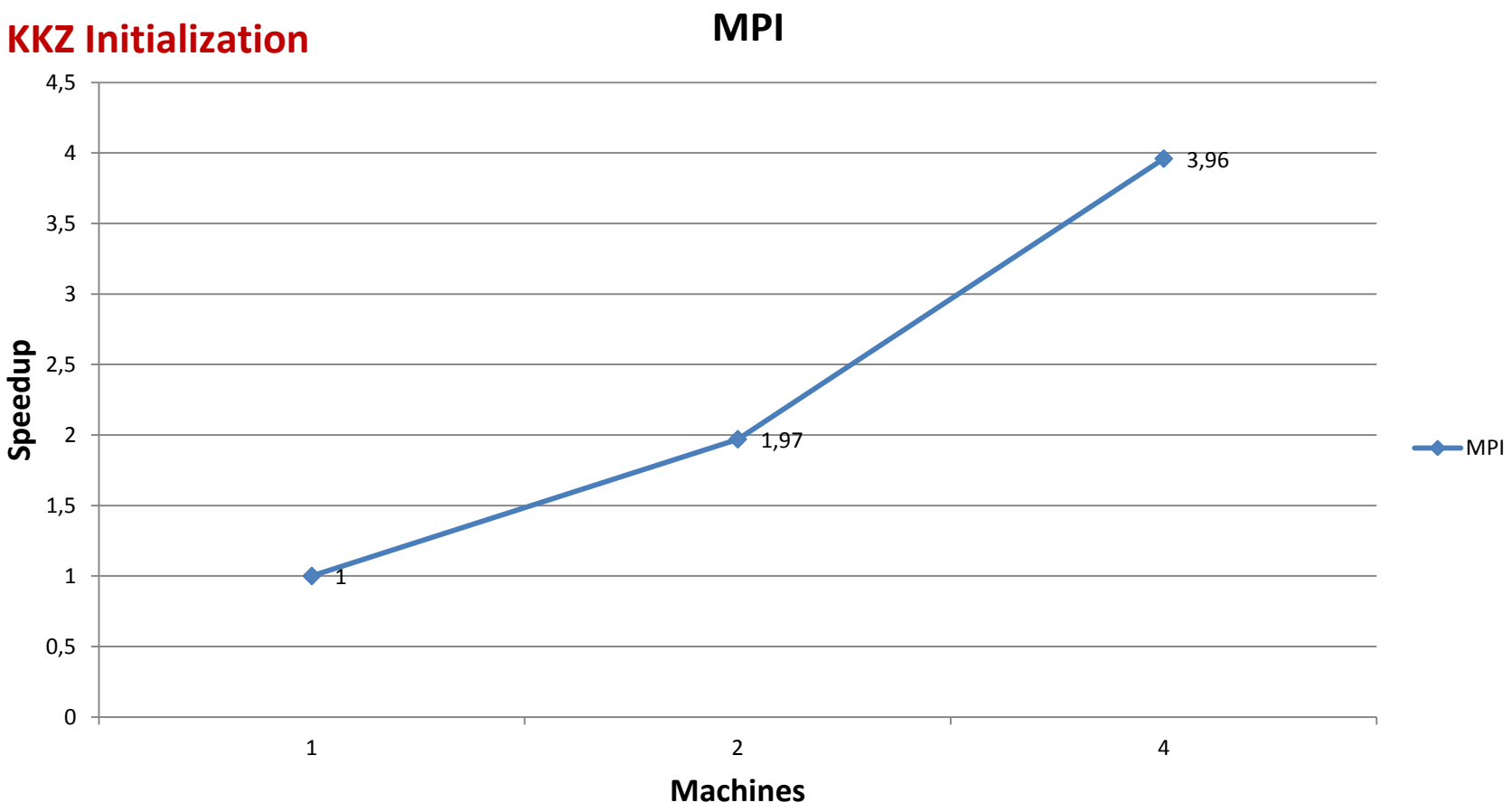
Χρόνοι εκτέλεσης - Hybrid vs. OpenMP vs. MPI vs. Serial

Random Initialization

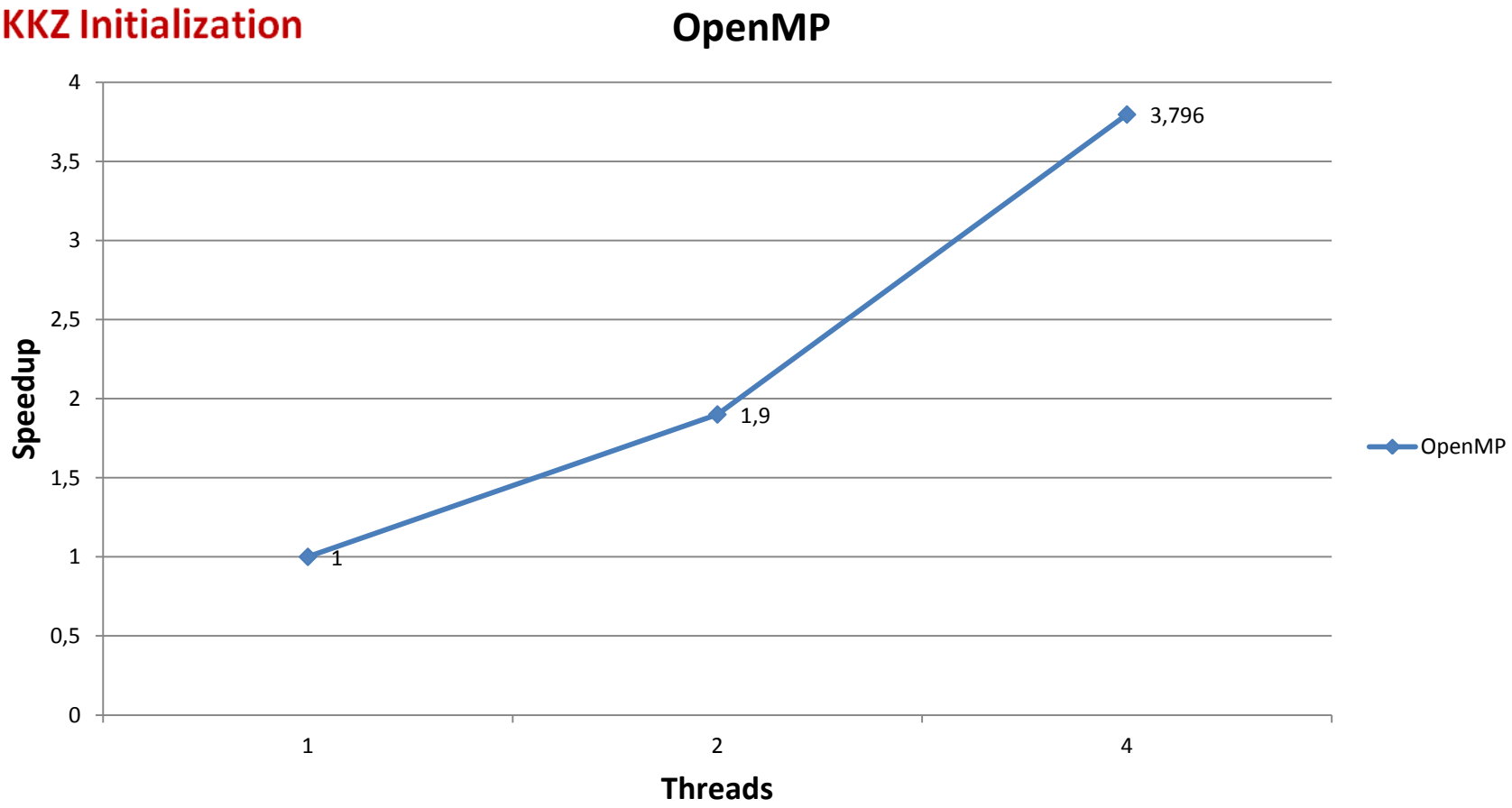


Pure MPI υλοποίηση- Speedup

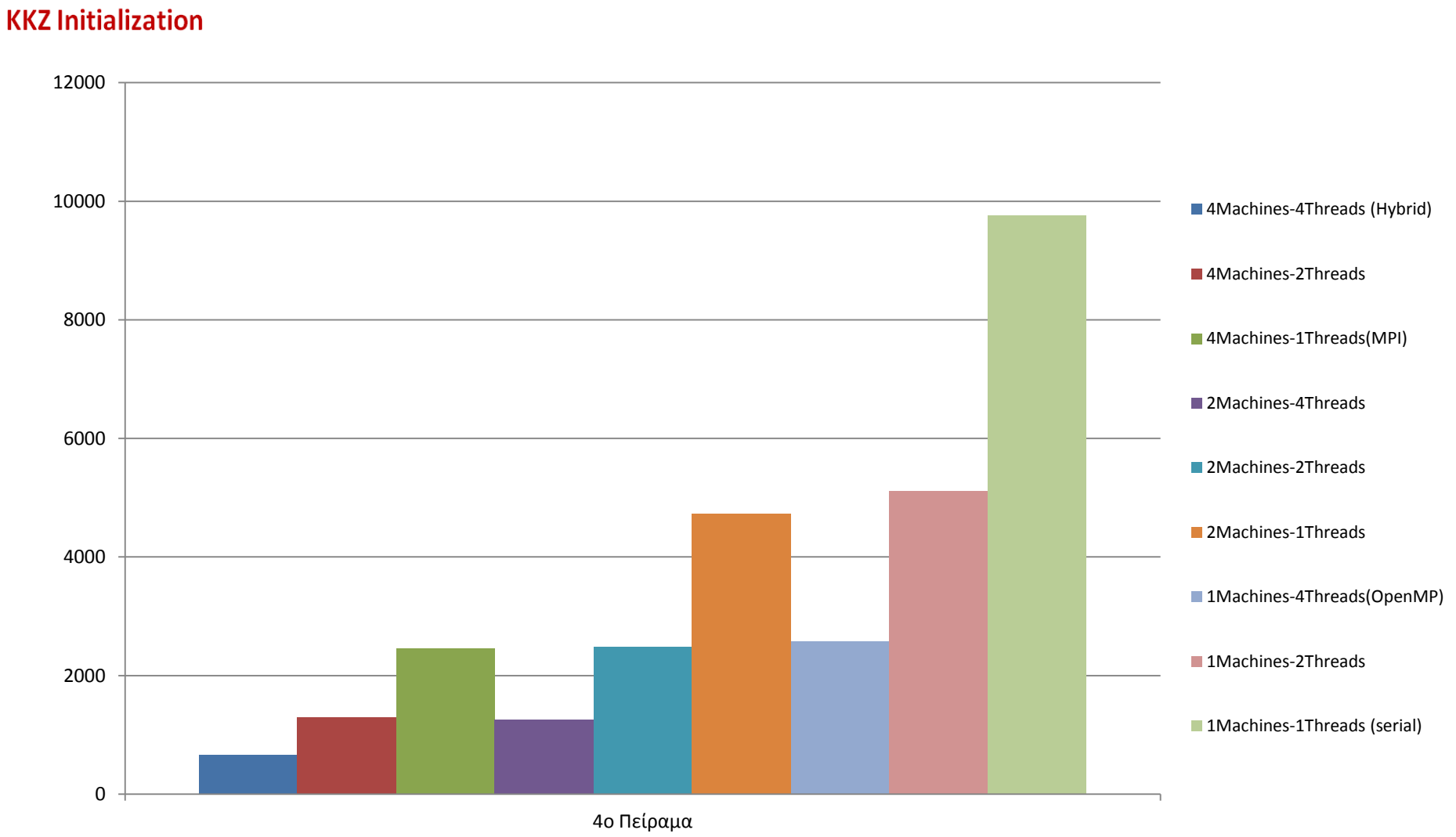
KKZ Initialization



Pure OpenMP υλοποίηση- Speedup

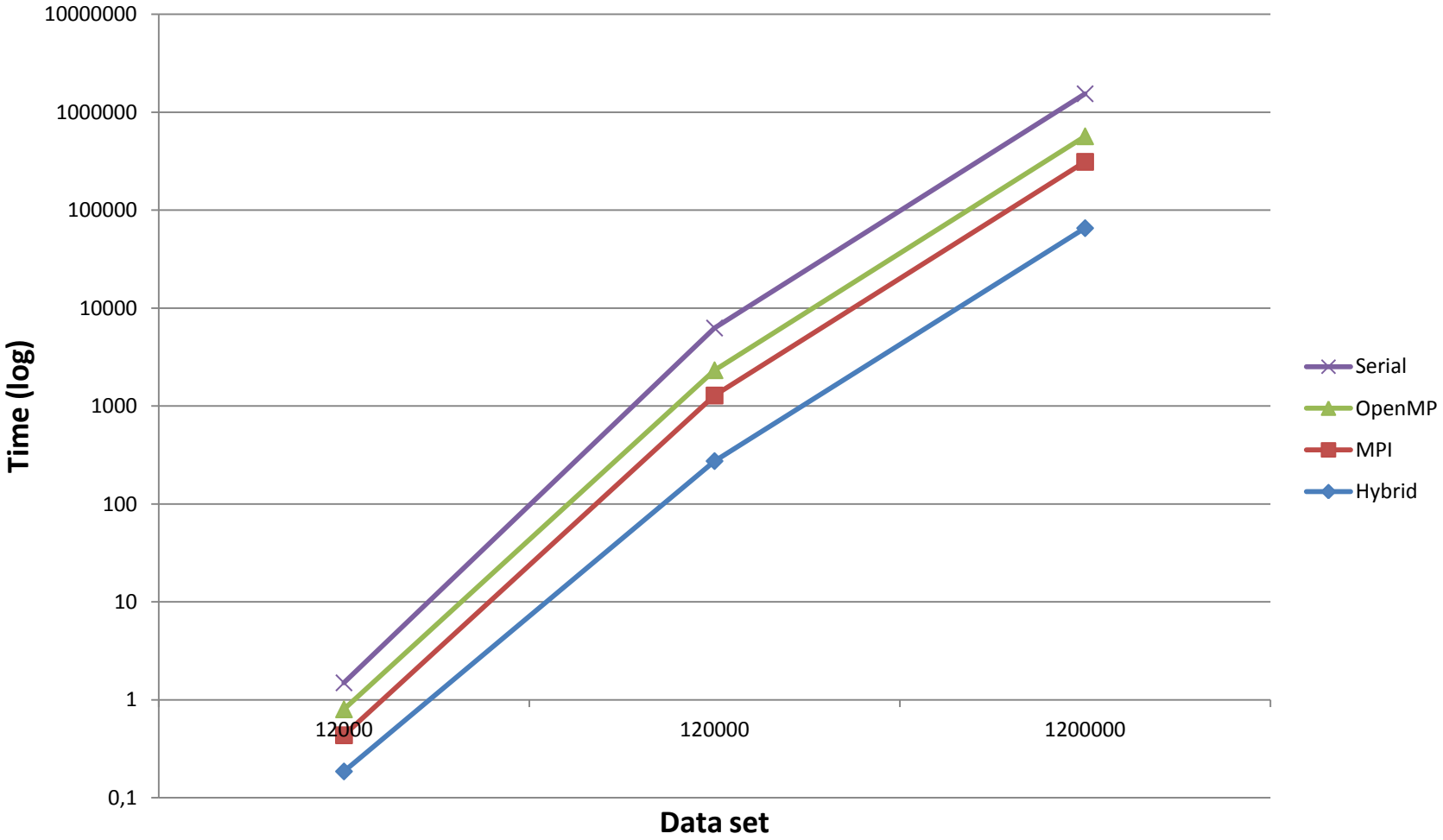


Αντικείμενα: 1.200.000, διάσταση: 16, κέντρα: 10.000



Χρόνοι εκτέλεσης - Hybrid vs. OpenMP vs. MPI vs. Serial

KKZ Initialization



KKZ vs. Random Initialization

Machines	Data Points	Dimension	Centroids	Threads	Random		KKZ	
					Iter	Time	Iter	Time
4	120000	16	1024	4	395	295,333141	370	274,259329
4	120000	16	1024	2	395	567,409306	370	531,497309
4	120000	16	1024	1	395	1074,647152	370	1006,631509
2	120000	16	1024	4	395	572,110406	330	477,024125
2	120000	16	1024	2	395	1121,669505	330	937,0909789
2	120000	16	1024	1	395	2126,563475	330	1776,62265
1	120000	16	1024	4	395	1128,838505	366	1040,770364
1	120000	16	1024	2	395	2226,06437	366	2062,631796
1	120000	16	1024	1	395	4250,252535	366	3938,208678
4	1200000	16	10000	4	999	69869,023594	937	65532,80736
4	1200000	16	10000	2	999	138176,8818	937	129600,5125
4	1200000	16	10000	1	999	262330,4804	937	246049,7058
2	1200000	16	10000	4	999	138229,3753	902	124807,688
2	1200000	16	10000	2	999	275509,7255	902	248757,8759
2	1200000	16	10000	1	999	523845,825	902	472981,712
1	1200000	16	10000	4	999	276057,2764	930	256990,2573
1	1200000	16	10000	2	999	549321,5985	930	511380,4668
1	1200000	16	10000	1	999	1048007,267	930	975622,1321

Συμπεράσματα

- Ο KKZ οδηγεί σε:
  - λιγότερες επαναλήψεις τον K-means άρα και σε μικρότερο χρόνο εκτέλεσης από την τυχαία αρχικοποίηση
  - Καλύτερη τοποθέτηση αντικειμένων από την τυχαία αρχικοποίηση

- Pure MPI γραμμικό speedup
- Pure OpenMP γραμμικό speedup
- MPI μικρή υπεροχή έναντι της OpenMP

- Hybrid υλοποίηση:
  - σαφής υπεροχή
  - και την κλιμάκωση μεταξύ των dataset

  τόσο των παράλληλων υλοποιήσεων όσο και της σειριακής

Ευχαριστούμε!

Ερωτήσεις;