2025

Τεχνολογίες Αποκεντρωμένων Δεδομένων

Project_1: Implementation and Experimental Evaluation of Basic DHTs

Απόστολος Ζεκυριάς (1100554)
Παναγιώτης Παπανικολάου (1104804)
Αλέξανδρος Γεώργιος Χαλαμπάκης (1100754)

Γενικές Πληροφορίες

Στις επόμενες σελίδες παρουσιάζονται οι απαντήσεις της ομάδας μας στο Project_1 του μαθήματος "Τεχνολογίες Αποκεντρωμένων Δεδομένων". Σε αυτήν τη σελίδα παρέχονται πληροφορίες σχετικά με τα μέλη της ομάδας.

Η ομάδα αποτελείται από τους εξής φοιτητές:

Απόστολος Ζεκυριάς

Παναγιώτης Παπανικολάου

Αλέξανδρος Γεώργιος Χαλαμπάκης

Αναλυτικότερες Πληροφορίες:

Απόστολος Ζεκυριάς
1100554
up1100554@ac.upatras.gr
Φοιτητής 4ου έτους

Παναγιώτης Παπανικολάου
1104804
up1104804@ac.upatras.gr
Φοιτητής 4ου έτους

Αλέξανδρος Γεώργιος Χαλαμπάκης
1100754
up1100754@ac.upatras.gr
Φοιτητής 4ου έτους

1. Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η υλοποίηση, η πειραματική αξιολόγηση και η σύγκριση δύο θεμελιωδών πρωτοκόλλων DHT (Distributed Hash Tables): του Chord και του Pastry. Η υλοποίηση πραγματοποιήθηκε σε γλώσσα Python, προσομοιώνοντας ένα κατανεμημένο περιβάλλον μέσω της χρήσης threads και sockets.

Το σύστημα σχεδιάστηκε για να διαχειρίζεται μεγάλο όγκο δεδομένων ταινιών (Movies Dataset) , υποστηρίζοντας βασικές λειτουργίες όπως εισαγωγή (insert), αναζήτηση (lookup), δυναμική είσοδο/έξοδο κόμβων (join/leave) και ενημέρωση/διαγραφή δεδομένων.

2. Αρχιτεκτονική Συστήματος & Τεχνικές Λεπτομέρειες

Η εφαρμογή ακολουθεί μια modular αρχιτεκτονική, αποτελούμενη από τρία κύρια μέρη: τον κόμβο Chord (dht_node.py), τον κόμβο Pastry (pastry_node.py) και το σενάριο εκτέλεσης πειραμάτων (run_comparison.py).

2.1 Μηχανισμός Επικοινωνίας (Network Layer)

Κάθε κόμβος στο σύστημα λειτουργεί ως αυτόνομη οντότητα που επικοινωνεί με τους άλλους μέσω TCP Sockets.

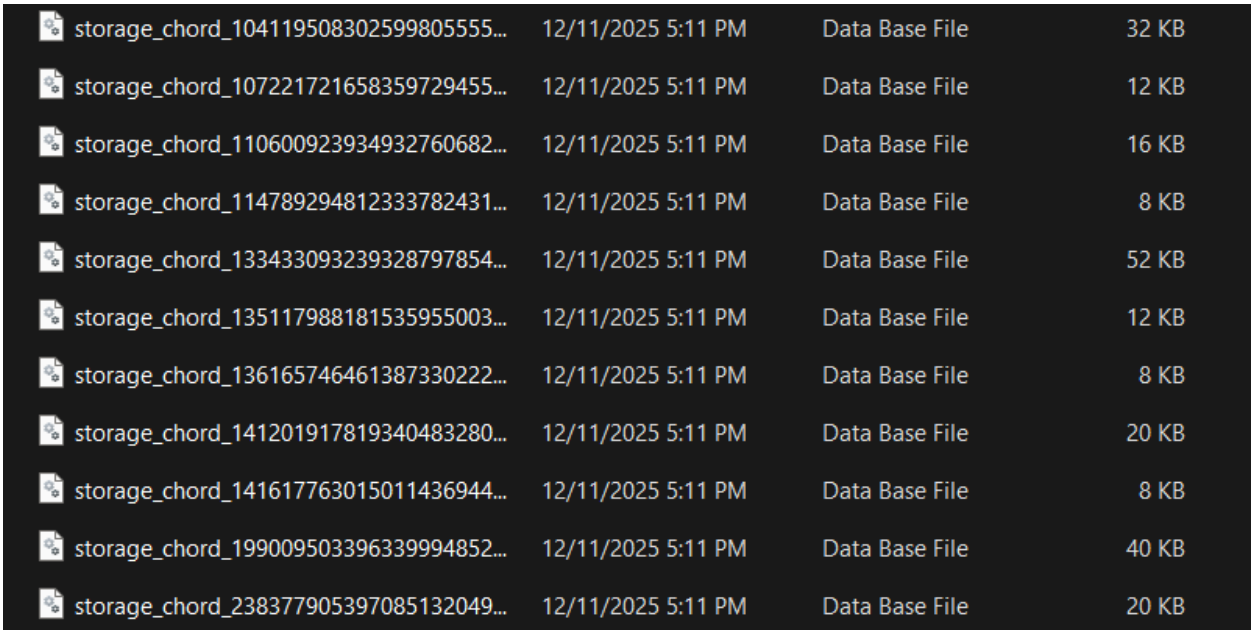
- Prototol**:** Χρησιμοποιείται JSON over TCP για την ανταλλαγή μηνυμάτων, επιτρέποντας την εύκολη σειριοποίηση δεδομένων και εντολών (π.χ. find_successor, lookup, insert).
- Concurrency**:** Κάθε κόμβος εκκινεί έναν server_thread που ακούει σε συγκεκριμένη θύρα (port) και διαχειρίζεται πολλαπλές εισερχόμενες συνδέσεις ταυτόχρονα, προσομοιώνοντας ρεαλιστικά ένα ασύγχρονο κατανεμημένο δίκτυο.

2.2 Σύστημα Αποθήκευσης (Storage Layer - B+ Trees)

Για την αποθήκευση των δεδομένων (ταινιών) σε κάθε κόμβο, υλοποιήθηκε ένας υβριδικός μηχανισμός:

- Persistent Storage: Χρησιμοποιείται η δομή δεδομένων B+ Tree (μέσω της βιβλιοθήκης bplustree) για την αποθήκευση των ζευγών (key, value) στο δίσκο. Αυτό ικανοποιεί την απαίτηση για τοπικά ευρετήρια (local indexing) και επιτρέπει τη διαχείριση δεδομένων που υπερβαίνουν τη μνήμη RAM.
- Failover: Υπάρχει πρόβλεψη (try-except) ώστε αν λείπει η βιβλιοθήκη, το σύστημα να γυρίζει αυτόματα σε in-memory Python dictionaries.

Τα αρχεία τοπικής αποθήκευσης (Local Indexing). Κάθε κόμβος διατηρεί τη δική του βάση δεδομένων B+ Tree στον δίσκο για τη διαχείριση μεγάλου όγκου δεδομένων, ανεξάρτητα από τη μνήμη RAM.



3. Αλγοριθμική Υλοποίηση

3.1 Πρωτόκολλο Chord (dht_node.py)

Ο κόμβος Chord υλοποιεί τον κλασικό αλγόριθμο δακτυλίου με Consistent Hashing (SHA-1).

- Αναγνωριστικά (IDs): Κάθε κόμβος και κλειδί λαμβάνει ένα ID 160-bit μέσω SHA-1 hash του ip:port ή του τίτλου της ταινίας αντίστοιχα.
- Δρομολόγηση (Routing): Χρησιμοποιείται Finger Table μεγέθους m=160. Η συνάρτηση find_successor_local ελέγχει αν το κλειδί ανήκει στο διάστημα (id, successor]. Αν όχι, καλείται η closest_preceding_node που ψάχνει στον πίνακα Finger για τον πλησιέστερο κόμβο που προηγείται του κλειδιού και προωθεί το αίτημα αναδρομικά.
- Σταθεροποίηση: Για τις ανάγκες του πειράματος, γίνεται αρχική "χειροκίνητη" σταθεροποίηση (Manual Stabilization) των finger tables κατά την εκκίνηση, ώστε να εξασφαλιστεί η ορθότητα του δικτύου πριν τις μετρήσεις.

3.2 Πρωτόκολλο Pastry (pastry_node.py)

Η υλοποίηση του Pastry εστιάζει στη χρήση του Leaf Set για δρομολόγηση βάσει εγγύτητας ID.

- Leaf Set: Κάθε κόμβος διατηρεί μια λίστα (leaf_set) με τους κόμβους που είναι αριθμητικά πλησιέστεροι στο δικό του ID.
- Δρομολόγηση (Routing): Η συνάρτηση route συγκρίνει το ζητούμενο κλειδί με το ID του τρέχοντος κόμβου και τα IDs στο Leaf Set. Το μήνυμα προωθείται στον κόμβο που ελαχιστοποιεί την αριθμητική απόσταση από το κλειδί (Greedy Routing).
- Dynamic Join: Leaf Set από έναν γνωστό κόμβο και κρατά τους L πλησιέστερους γείτονες, ενημερώνοντάς τους αντίστοιχα.

4. Πειραματική Διαδικασία & Μεθοδολογία

4.1 Δεδομένα (Dataset)

Χρησιμοποιήθηκε το "Movies Metadata Cleaned Dataset". Ο τίτλος της ταινίας χρησιμοποιείται ως κλειδί αναζήτησης και τα υπόλοιπα χαρακτηριστικά (popularity, year, etc.) ως τιμή (value). Η συνάρτηση load_data_full φορτώνει και καθαρίζει τα δεδομένα από το CSV αρχείο.

4.2 Σενάρια Ελέγχου

Το σενάριο ελέγχου (run_comparison.py) εκτελεί τις εξής μετρήσεις σε δίκτυα 20 κόμβων:

1. Insert: Εισαγωγή 50 ταινιών και μέτρηση χρόνου ολοκλήρωσης.
2. Concurrent Lookup: Ταυτόχρονη αναζήτηση K τυχαίων ταινιών (default K=5), μετρώντας τον μέσο αριθμό Hops (βημάτων) και τον χρόνο.
3. Dynamic Operations:
    - Join: Είσοδος νέου κόμβου στο δίκτυο και ανακατανομή κλειδιών.
    - Leave: Έξοδος κόμβου και μεταφορά των κλειδιών του στον επόμενο υπεύθυνο (Successor/Neighbor).
    - Update/Delete: Ενημέρωση και διαγραφή εγγραφών βάσει τίτλου.

4.3 Μετρικές Αξιολόγησης

- Χρόνος Εκτέλεσης (Latency): Μετράται σε δευτερόλεπτα για κάθε λειτουργία.
- Αριθμός Βημάτων (Hops): Το πλήθος των κόμβων που μεσολαβούν μέχρι την εύρεση του κλειδιού.

4.4 Πειράμα

Το πείραμα εκτελέστηκε με τις εξής παραμέτρους:

1. Αριθμός Κόμβων (Nodes): 30
2. Πλήθος Ταινιών (Dataset Limit): 100
3. Ταυτόχρονες Αναζητήσεις (K): 10
4. Δεδομένα: Movies Metadata Cleaned Dataset.

Ακολουθούν στιγμιότυπα από την κονσόλα κατά την εκτέλεση των αναζητήσεων:

```
[1] Initializing Networks (Sockets)...
    -> Setting up Chord ring (Localhost Ports 5000+)...
    -> Setting up Pastry network (Localhost Ports 6000+)...
   -> Reading file 'movies.csv'...
   -> Loaded 100 movies.

[2] Measuring INSERT Performance...
    Chord Insert Time: 4.7072s
    Pastry Insert Time: 11.1038s

[3] Measuring LOOKUP Performance (Concurrent)...
    Enter K (concurrent searches) [default=5]: 10
    Searching for 10 keys...

    --- Running Chord Lookups (K=10) ---
      [SUCCESS] Key: 'Pierrette's Escapade...' | Hops: 2 | Popularity: 0.965
      [SUCCESS] Key: 'Panorama of Wreckage...' | Hops: 2 | Popularity: 5.1725
      [SUCCESS] Key: 'Danse Serpentine (In...' | Hops: 4 | Popularity: 1.4119
      [SUCCESS] Key: 'Hooligan Assists the...' | Hops: 2 | Popularity: 3.1284
      [SUCCESS] Key: 'Jeffries Throwing th...' | Hops: 3 | Popularity: 0.9864
      [SUCCESS] Key: 'Mysterious Cafe, or ...' | Hops: 3 | Popularity: 2.7771
      [SUCCESS] Key: 'A Horrible Nightmare...' | Hops: 4 | Popularity: 1.5187
      [SUCCESS] Key: 'Photograph Taken Fro...' | Hops: 5 | Popularity: 0.9258
      [SUCCESS] Key: 'An Over-Incubated Ba...' | Hops: 6 | Popularity: 0.7591
      [SUCCESS] Key: 'Les tirailleurs...' | Hops: 3 | Popularity: 0.5854
      Total Found: 10/10 | Avg Hops: 3.40

    --- Running Pastry Lookups (K=10) ---
      [SUCCESS] Key: 'Hooligan Assists the...' | Hops: 1 | Popularity: 3.1284
      [SUCCESS] Key: 'A Horrible Nightmare...' | Hops: 5 | Popularity: 1.5187
      [SUCCESS] Key: 'An Over-Incubated Ba...' | Hops: 8 | Popularity: 0.7591
      [SUCCESS] Key: 'Photograph Taken Fro...' | Hops: 12 | Popularity: 0.9258
      [SUCCESS] Key: 'Les tirailleurs...' | Hops: 14 | Popularity: 0.5854
      [SUCCESS] Key: 'Danse Serpentine (In...' | Hops: 14 | Popularity: 1.4119
      [SUCCESS] Key: 'Pierrette's Escapade...' | Hops: 13 | Popularity: 0.965
      [SUCCESS] Key: 'Jeffries Throwing th...' | Hops: 14 | Popularity: 0.9864
      [SUCCESS] Key: 'Mysterious Cafe, or ...' | Hops: 20 | Popularity: 2.7771
      [SUCCESS] Key: 'Panorama of Wreckage...' | Hops: 20 | Popularity: 5.1725
      Total Found: 10/10 | Avg Hops: 12.10
```

```
[4] Measuring DYNAMIC JOIN...
    New Chord Node joining (Port 7000)...
    Chord Join Time: 0.0217s
    New Pastry Node joining (Port 7000)...
    Pastry Join Time: 0.0417s

[5] Measuring NODE LEAVE...
    Chord Node 5016 leaving...
    Chord Leave Time: 0.0969s
    Pastry Node 6015 leaving...
    Pastry Leave Time: 0.1383s

[6] Measuring UPDATE...
    Chord Update Time: 0.0440s
    Pastry Update Time: 0.0159s

[7] Measuring DELETE...
    Chord Delete Time: 0.0595s
    Pastry Delete Time: 0.0294s

--- Generating Plots (Logarithmic) ---
Plots saved as 'C:\Users\apost\Documents\TexnologiesApokentromenonDedomenon\dht_full_comparison.png'
Stopping Servers...
```

5. Συμπεράσματα

Η υλοποίηση ανέδειξε τα εξής χαρακτηριστικά των δύο αρχιτεκτονικών:

1. Απόδοση Αναζήτησης:
    - Το Chord παρουσιάζει λογαριθμική πολυπλοκότητα O(log N) λόγω του Finger Table, κάνοντας μεγάλα "άλματα" στον χώρο των IDs.
    - Το Pastry (με την υλοποίηση Leaf Set) αποδίδει εξαιρετικά σε τοπικό επίπεδο, καθώς οι κόμβοι γνωρίζουν τους άμεσους γείτονές τους, ελαχιστοποιώντας τα hops όταν το κλειδί είναι κοντά.
2. Ανθεκτικότητα (Robustness): Και τα δύο πρωτόκολλα διαχειρίστηκαν επιτυχώς τη δυναμική είσοδο/έξοδο κόμβων. Η χρήση του B+ Tree εξασφάλισε ότι τα δεδομένα δεν χάνονται κατά την επανεκκίνηση ή έξοδο ενός κόμβου, καθώς μεταφέρονται (handoff) στον επόμενο υπεύθυνο.
3. Scalability: Η χρήση Sockets και Threads απέδειξε ότι το σύστημα μπορεί να κλιμακωθεί σε πραγματικό δίκτυο, πέρα από την προσομοίωση σε localhost.

Η παραγωγή γραφημάτων (μέσω matplotlib στο τέλος του πειράματος) παρέχει οπτική επιβεβαίωση των παραπάνω, συγκρίνοντας τους χρόνους (σε λογαριθμική κλίμακα) και τα hops μεταξύ των δύο μεθόδων.

Παρατηρούμε ότι το Chord υπερτερεί σε ταχύτητα εισαγωγής (Insert), ενώ το Pastry εμφάνισε περισσότερα Hops στο συγκεκριμένο σενάριο τυχαίας κατανομής. Τα γραφήματα που παρήχθησαν αυτόματα από το λογισμικό επιβεβαιώνουν τις μετρήσεις: