Σύγκριση της Απόδοσης της Neo4j Χρησιμοποιώντας Μεγάλου Όγκου Δεδομένα ενός Απλού Κοινωνικού Δικτύου

Για το μάθημα
Διαχείριση Μεγάλου Όγκου Δεδομένων στο Διαδίκτυο

Γιώργος Γεραρχάκης
Τμήμα Πληροφορικής
Ιόνιο Πανεπιστήμιο
Κέρκυρα, 49100, Ελλάδα
p17gera@ionio.gr

Πασχάλης Γρίβας
Τμήμα Πληροφορικής
Ιόνιο Πανεπιστήμιο
Κέρκυρα, 49100, Ελλάδα
p17griv@ionio.gr

Περίληψη

Η Neo4j είναι μία NoSQL βάση δεδομένων γράφων ανοιχτού κώδικα. Η βάση, συνδέει τα δεδομένα καθώς αυτά αποθηκεύονται επιτρέποντας την εκτέλεση σύνθετων ερωτημάτων σε ελάχιστο χρόνο. Σκοπός της εργασίας είναι να μετρηθεί η απόδοση της βάσης, κατά τη μαζική εισαγωγή δεδομένων ενός απλού κοινωνικού δικτύου και κατά την εκτέλεση ερωτημάτων, υπό διάφορες συνθήκες, όπως διαφορετικό όγκο δεδομένων και διαφορετική πολυπλοκότητα ερωτημάτων.

1 Εισαγωγή

Η Neo4j είναι μία NoSQL βάση δεδομένων γράφων ανοιχτού κώδικα. Η ανάπτυξή της ξεκίνησε το 2003 ενώ η πρώτη διαθέσιμη για το κοινό έκδοσή της δημοσιεύτηκε το 2007. Ο πηγαίος κώδικάς της είναι γραμμένος σε Java και Scala. Η βάση δεδομένων διατίθεται σε δύο εκδόσεις, την Community Edition, στην ανάπτυξη της οποία συνεισφέρει η κοινότητά της, και η Enterprise Edition, η οποία παρέχει επιπλέων λειτουργικότητες όπως, δημιουργία αντιγράφων ασφαλείας (backups), clustering κ.α..

Η Neo4j υλοποιεί το μοντέλο γράφου με ιδιότητες (οι σχέσεις δεν είναι απλώς συνδέσεις αλλά περιέχουν τύπο και κάποιες ιδιότητες - property graph model) έως το επίπεδο αποθήκευσης των δεδομένων. Αυτό σημαίνει ότι τα δεδομένα αποθηκεύονται με βάση αυτό το μοντέλο και η βάση χρησιμοποιεί δείκτες προκειμένου να διασχίσει τους γράφους.

Η Neo4j, ως τρόπο αλληλεπίδρασης με τα δεδομένα, παρέχει τη γλώσσα Cypher, μία δηλωτική γλώσσα ερωτημάτων (query language) βασισμένη στην SQL. Η γλώσσα, επιτρέπει στους χρήστες να αποθηκεύουν, ενημερώνουν και ανακτούν δεδομένα από τη βάση δεδομένων. Η σύνταξή της, παρέχει έναν οπτικό και λογικό τρόπο εύρεσης μοτίβων κόμβων και σχέσεων στους γράφους.

Ένας άλλος τρόπος αλληλεπίδρασης με τα δεδομένα των Neo4j βάσεων, είναι μέσω των οδηγών (drivers), οι οποίοι παρέχονται και οι οποίοι επιτρέπουν την ανάπτυξη εφαρμογών, σε διάφορες γλώσσες προγραμματισμού. Οι γλώσσες, για τις οποίες επισήμως παρέχονται οδηγοί είναι, η .Net, η Java, η JavaScript και η Python. Οι οδηγοί αυτοί, χρησιμοποιούν το δυαδικό πρωτόκολλο μεταφοράς δεδομένων “Bolt”. Τέλος, υπάρχουν υλοποιημένοι από την κοινότητα της Neo4j, οδηγοί και για άλλες γλώσσες προγραμματισμού, για το δυαδικό πρωτόκολλο “Bolt” και το πρωτόκολλο “Http”.

Βασικό χαρακτηριστικό της Neo4j βάσης δεδομένων, είναι οι σταθεροί χρόνοι των διασχίσεων σε μεγάλους γράφους, είτε κατά βάθος είτε κατά πλάτος, λόγω της αποδοτικής αναπαράστασης των κόμβων και των σχέσεων. Το γεγονός αυτό, επι-

τρέπει επιτρέπει την εύκολη διαχείριση γράφων με μεγάλο αριθμό κόμβων σε μέτριας απόδοσης συστήματα.

Σκοπός της εργασίας, είναι να ερευνηθεί η απόδοση μίας Neo4j βάσης δεδομένων, όταν αυτή υποβάλλεται σε διαφορετικές συνθήκες. Πιο συγκεκριμένα, ερευνάται ο χρόνος μαζικής εισαγωγής δεδομένων ενός απλού κοινωνικού δικτύου, μέσω του παρεχόμενου από τη Neo4j εργαλείου "neo4j-admin import" και συγκρίνεται σε διαφορετικού όγκου δεδομένα – αριθμό χρηστών και σε διαφορετικά υπολογιστικά συστήματα. Επίσης, ερευνητικό αντικείμενο της εργασίας, είναι ο χρόνος, ο οποίος απαιτείται, ώστε να εκτελεστούν ερωτήματα, διαφόρων πολυπλοκοτήτων (απλά, μεσαία, σύνθετα), σε Neo4j βάσεις με διαφορετικό πλήθος κόμβων και σχέσεων.

2 Μεθοδολογία

Προκειμένου να ερευνηθεί η απόδοση της Neo4j, εγκαταστάθηκε το "Neo4j Desktop" με την έκδοση 4.2.3 της βάσης δεδομένων Neo4j, σε λειτουργικό σύστημα Windows 10 x64. Επίσης, για την εκτέλεση των προγραμμάτων, τα οποία αλληλεπιδρούν με την βάση προκειμένου να μετρηθεί η απόδοσή της, χρησιμοποιήθηκε η έκδοση 3.8.5 της Python και ο επίσημος οδηγός (driver), ο οποίος παρέχεται για τη γλώσσα.

2.1 Παραγωγή Δεδομένων

Για την παραγωγή των δεδομένων, αρχικά συντάχθηκε σε Python (αρχείο *generate_users.py*) ένα πρόγραμμα, το οποίο παράγει πληροφορίες για ένα δεδομένο αριθμό χρηστών. Πιο συγκεκριμένα, για κάθε χρήστη παράγονται τιμές (τυχαίες ή επιλέγονται τυχαία από κάποιο dataset) για τα πεδία ονοματεπώνυμο, φύλο, ημερομηνία γέννησης, email, τηλέφωνο, χώρα και πόλη. Οι πληροφορίες αυτές, εξάγονται σε ένα αρχείο CSV με όνομα σχετικό με το πλήθος των χρηστών, οι οποίοι παράχθηκαν.

Προκειμένου να παραχθούν σχέσεις μεταξύ των χρηστών, αναπτύχθηκε πρόγραμμα Python (αρχείο *generate_relationships.py*), το οποίο δεδομένου ενός πλήθους χρηστών, δημιουργεί για κάθε χρήστη 100 σχέσεις (τύπου ο χρήστης Α ακολουθεί το χρήστη Β" με άλλους τυχαίους χρήστες του πλήθους. Οι σχέσεις αυτές, αποθηκεύονται σε ένα αρχείο CSV με όνομα σχετικό με το πλήθος των χρηστών, για τους οποίους παράχθηκαν σχέσεις.

Τέλος, τα δύο αυτά προγράμματα εκτελέστηκαν προκειμένου να δημιουργηθούν αρχεία με 100 χιλιάδες, 500 χιλιάδες, 1 εκατομμύριο και 5 εκατομμύρια χρήστες και τα αντίστοιχα αρχεία με σχέσεις.

2.2 Εισαγωγή Δεδομένων

Προκειμένου να εισαχθούν μαζικά (batch import) τα δεδομένα, τα οποία παράχθηκαν, αρχικά δημιουργήθηκαν 4 διαφορετικά instances – γράφοι, ένας για κάθε πλήθος δεδομένων. Στη συνέχεια, σε κάθε instance, χρησιμοποιήθηκε το εργαλείο "neo4j-admin import" (Bulk Importer) προκειμένου να εισαχθούν στους γράφους οι χρήστες ως κόμβοι και οι σχέσεις των χρηστών ως ακμές.

Το εργαλείο "neo4j-admin import", χρησιμοποιείται για την φόρτωση μεγάλου όγκου δεδομένων, από CSV αρχεία, σε μία άδεια Neo4j βάση. Επομένως, το εργαλείο είναι ιδανικό μόνο για την αρχικοποίηση ενός γράφου με έναν αρχικό πληθυσμό. Έτσι, ως είσοδο δέχεται ένα (ή περισσότερα) αρχείο CSV, το οποίο περιλαμβάνει τους κόμβους του γράφου, με τις ιδιότητές τους, και ένα άλλο αρχείο CSV, το οποίο περιέχει τις σχέσεις – συνδέσεις των κόμβων.

Η διαδικασία εισαγωγής των δεδομένων από τα αχρεία χωρίζεται, από το εργαλείο αυτό, σε 4 στάδια. Στο πρώτο στάδιο, εισάγονται στη βάση οι κόμβοι ενώ στο δεύτερο εισάγονται οι σχέσεις. Στο τρίτο στάδιο, γίνεται η σύνδεση των σχέσεων με του κόμβους, βάση του Id τους και τέλος στο τέταρτο βήμα γίνεται post processing των δεδομένων της βάσης.

Επομένως, σε κάθε instance η εντολή, η οποία εκτελέστηκε ήταν η εξής:

```
bin\neo4j-admin import
--database=neo4j
--delimiter=","
--nodes=import/users.csv
--relationships=import/relation-
ships.csv
--skip-bad-relationships=true
```

Η διαδικασία έγινε σε 2 διαφορετικά συστήματα, ένα μέτριας (Intel i7 3770 – 4 cores, 8 threads) και ένα καλής επεξεργαστικής ισχύος (AMD Ryzen 7 3700X – 8 cores, 16 threads). Και στις δύο περιπτώσεις, τα συστήματα χρησιμοποίησαν ίδιους

SSD δίσκους “Samsung SSD 860 EVO 500GB” και χρησιμοποιήθηκαν οι εξ ορισμού ρυθμίσεις για το μέγεθος της στοίβας στην κύρια μνήμη:

```
dbms.memory.heap.initial_size=512m
dbms.memory.heap.max_size=1G
```

Τέλος, σε κάθε εκτέλεση του εργαλείου, σε κάθε instance, έγινε καταγραφή των χρόνων εισαγωγής.

2.3 Εκτέλεση Ερωτημάτων

Σε ότι αφορά τη μέτρηση της απόδοσης της Neo4j στις βάσεις, οι οποίες δημιουργήθηκαν και γέμισαν με δεδομένα, όπως περιγράφηκε στην προηγούμενη ενότητα, αναπτύχθηκε ένα ακόμη βοηθητικό πρόγραμμα σε Python (αρχείο query.*py*). Μέσω του προγράμματος αυτού, εκτελέστηκαν τα εξής τρία ερωτήματα (queries) κλιμακούμενης πολυπλοκότητας (από το πιο απλό προς το πιο σύνθετο):

1. `MATCH (n:User {ctry:"United Kingdom"})RETURN n`
2. `MATCH (n:User {ctry:"United Kingdom"})-[:FOLLOWS]->(m:User {ctry:"United Kingdom"}) RETURN n,m`
3. `MATCH (n:User {ctry:"United Kingdom"})-[:FOLLOWS]->(m:User {ctry:"United Kingdom"})-[:FOLLOWS]->(l:User {ctry:"United Kingdom"})-[:FOLLOWS]->(n) RETURN n,m,l`

Το πρώτο ερώτημα (απλό), αφορά την εύρεση χρηστών, οι οποίοι βρίσκονται στο Ηνωμένο Βασίλειο (η τιμή του “ctry” property να είναι ίση με “United Kingdom”). Στο δεύτερο ερώτημα (μέτριο), ζητείται από τη βάση να επιστρέψει για κάθε χρήστη από το Ηνωμένο Βασίλειο τους χρήστες από το Ηνωμένο Βασίλειο, τους οποίους ακολουθεί. Τέλος, στο τρίτο ερώτημα (σύνθετο), ζητείται από τη βάση να βρει και να επιστρέψει για κάθε χρήστη από το Ηνωμένο Βασίλειο, χρήστες από το Ηνωμένο Βασίλειο, τους οποίους ακολουθεί και οι οποίοι ακολουθούν χρήστες από το Ηνωμένο Βασίλειο, οι οποίοι ακολουθούν τους πρώτους (τρίγωνα).

Το πρόγραμμα εκτελεί σε κάθε instance, με διαφορετικό όγκο δεδομένων, τα παραπάνω τρία ερωτήματα και καταγράφει κάθε φορά το χρόνο απόκρισης της βάσης στην πρώτη εκτέλεση του κάθε ερωτήματος, δηλαδή χωρίς να έχει δημιουργηθεί και αποθηκευτεί το query plan στη μνήμη (cold cache). Στη συνέχεια, το πρόγραμμα εκτελεί κάθε ερώτηση 10 φορές καταγράφοντας κάθε φορά τον χρόνο εκτέλεσης και εξάγει το μέσο χρόνο εκτέλεσης.

Τέλος, η διαδικασία αυτή επαναλήφθηκε μία ακόμη φορά, αφού είχε προηγηθεί η δημιουργία ευρετηρίων (single-property indexes), σε κάθε instance, στο στην ιδιότητα (property) “ctry”, καθώς αυτή χρησιμοποιείται από τα ερωτήματα.

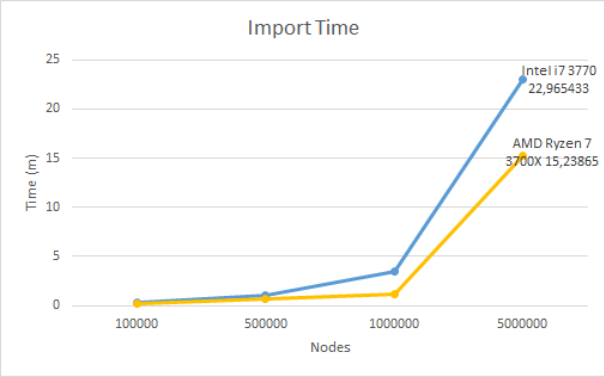
3 Αποτελέσματα

3.1 Εισαγωγή Δεδομένων

Από τους απαιτούμενους χρόνους του εργαλείου “neo4j-admin import”, για την εισαγωγή των δεδομένων από τα αρχεία CSV σε κάθε instance της Neo4j, προκύπτει ότι όσο αυξάνεται ο όγκος των δεδομένων – πλήθος των χρηστών και των σχέσεων, τόσο μεγαλύτερος είναι ο χρόνος, ο οποίος απαιτείται. Επίσης, διαφορά υπάρχει και στους χρόνους εισαγωγής των δεδομένων ανάμεσα στα δυο συστήματα. Επιπλέον, όπως φαίνεται και στο *διάγραμμα 1* ενδιαφέρον παρουσιάζει η μεγάλη απόσταση μεταξύ της διαφοράς του χρόνου, ο οποίος απαιτείται για την εισαγωγή των 100 χιλιάδων και των 500 χιλιάδων, με την διαφορά του χρόνου, ο οποίος απαιτείται για την εισαγωγή των 1 εκατομμυρίου και 5 εκατομμυρίων κόμβων. Αναλυτικά οι καταγεγραμμένοι χρόνοι φαίνονται στον παρακάτω πίνακα:

	Time (m)	
Nodes	**Intel i7 3770**	**AMD Ryzen 7 3700X**
100000	0,25	0,19
500000	1,04	0,64
1000000	3,44	1,2
5000000	22,97	15,24

Πίνακας 1: Χρόνοι του Import Tool

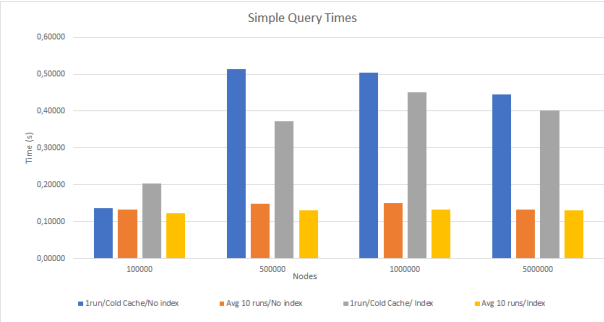


Διάγραμμα 1: Σύγκριση των χρόνων του Import Tool

3.2 Εκτέλεση Ερωτημάτων

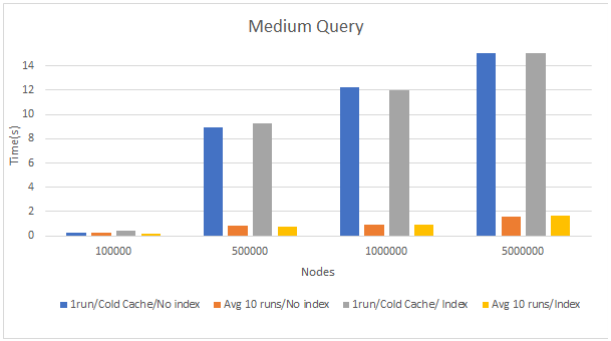
Σύμφωνα με τα αποτελέσματα του βοηθητικού προγράμματος εκτέλεσης ερωτημάτων, το οποίο εκτελέστηκε σε κάθε instance της βάσης, οι χρόνοι για την εκτέλεση των ερωτημάτων διαφέρουν ως προς τη χρήση ή όχι ευρετηρίου, το πλήθος των κόμβων της βάσης και το αν έχει γίνει caching των ερωτημάτων ή όχι.

Στους χρόνους εκτέλεσης του πρώτου ερωτήματος (απλού ερωτήματος – *διάγραμμα 2*), παρατηρείται το γεγονός ότι όσο μεγαλώνει το πλήθος της βάσης, αυτοί παρουσιάζουν μικρές διαφορές. Επίσης, παρατηρείται διαφορά ανάμεσα στις πρώτες εκτελέσεις του ερωτήματος (cold cache), ανάμεσα στις περιπτώσεις όπου έχει δημιουργηθεί ευρετήριο (μπλε στήλες) στη βάση και σε αυτές όπου δεν έχει δημιουργηθεί (γκρι στήλες).



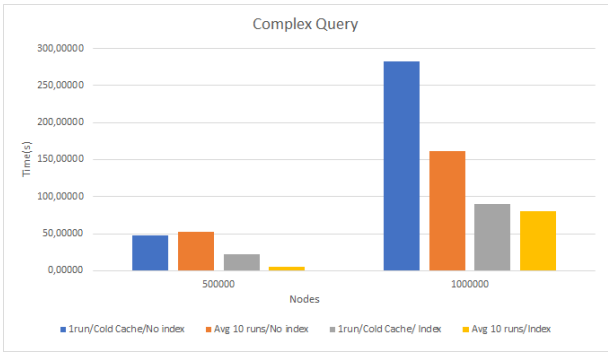
Διάγραμμα 2: Χρόνοι εκτέλεσης του "Απλού Ερωτήματος"

Σχετικά με τους χρόνους εκτέλεσης του δεύτερου ερωτήματος, (μεσαίου ερωτήματος – *διάγραμμα 3*), αρχικά παρατηρείται μεγάλη αύξησή τους από την περίπτωση, στην οποία εκτελείται στο instance με τους 100 χιλιάδες κόμβους, με τις περιπτώσεις όπου εκτελείται στα instances μεγαλύτερων πληθυσμών. Επίσης, σε αντίθεση με τους χρόνους εκτέλεσης του απλού ερωτήματος, εδώ δεν παρατηρείται η διαφορά (μείωση) του χρόνου εκτέλεσης ανάμεσα στις περιπτώσεις όπου έχει δημιουργηθεί ευρετήριο (μπλε στήλες) στη βάση και σε αυτές όπου δεν έχει δημιουργηθεί (γκρι στήλες), στις πρώτες εκτελέσεις του ερωτήματος (cold cache).



Διάγραμμα 3: Χρόνοι εκτέλεσης του "Μεσαίου Ερωτήματος"

Οι χρόνοι εκτέλεσης του τρίτου ερωτήματος, στο instance με πλήθος 100 χιλιάδες χρήστες πραγματοποιήθηκε σε εξίσου μικρούς χρόνους με τις εκτελέσεις των άλλων ερωτημάτων στο ίδιο instance. Επίσης, το ερώτημα αυτό, στο instance με πλήθος 5 εκατομμυρίων χρηστών, πήρε πολύ περισσότερο χρόνο (2h+) σε σχέση με τα άλλα ερωτήματα σε αυτό το instance και για τον λόγο αυτό δε μπορεί να συγκριθεί. Τέλος, αυτό το οποίο μπορεί να παρατηρηθεί στους χρόνους αυτού του ερωτήματος είναι, η μείωση των χρόνων εκτέλεσης στις περιπτώσεις όπου έχει προηγηθεί η δημιουργία ευρετηρίου, τόσο στις πρώτες εκτελέσεις όσο και στις επόμενες (cached) εκτελέσεις.



Διάγραμμα 4: Χρόνοι εκτέλεσης του "Σύνθετου Ερωτήματος"

Τέλος, οι παρακάτω πίνακες (*πίνακας 2 & 3*) συγκεντρώνουν όλους τους χρόνους εκτέλεσης και των τριών ερωτημάτων.

1st run/Cold Cache			Avg 10 runs		
simple	mediu m	com- plex	simple	mediu m	com- plex
1,05	17,36	6,42	0,05	0,09	0,26
0,89	32,70	65,65	0,05	0,43	6,58
0,86	34,65	157,10	0,05	0,51	16,31
2,15	62,63	-	0,05	0,99	-

Πίνακας 2: Χρόνοι εκτέλεσης των ερωτημάτων – ***No index***

1st run/Cold Cache			Avg 10 runs		
simple	mediu m	com- plex	simple	mediu m	com- plex
0,18	0,22	0,42	0,06	0,10	0,22
0,17	6,72	15,18	0,05	0,41	3,30
0,17	8,54	47,75	0,05	0,49	8,17
0,17	10,99	-	0,05	0,95	-

Πίνακας 3: Χρόνοι εκτέλεσης των ερωτημάτων – ***With index***