**Information Systems, Analysis and Design**

**Comparison between Time Series DataBases**

|  |  |  |
| --- | --- | --- |
| Λαουρεντιάν Γκούμε | Μαρία Τσιγάρα | Κρις Κούτση |
| (Α.Μ. 03118014) | (Α.Μ. 03118823) | (Α.Μ. 03118905) |

# Εισαγωγή

Η παρούσα εργασία περιλαμβάνει την εγκατάσταση και ρύθμιση των InfluxDB και TimescaleDB, τη δημιουργία και τη φόρτωση ενός μεγάλου όγκου Time Series δεδομένων σε κάθε σύστημα, τη δημιουργία ερωτημάτων με σκοπό τον έλεγχο της απόδοσης, και τη μέτρηση μετρικών απόδοσης για τη σύγκριση των δύο συστημάτων. Μάλιστα, η συγκριτική ανάλυση ανάμεσα στα δύο δημοφιλή αυτά συστήματα βάσεων δεδομένων αποτελεί ουσιαστικά και τον σκοπό της εργασίας μας, καθώς μέσω αυτής της ανασκόπησης θα μπορέσουμε να εμβαθύνουμε και τελικώς να κατανοήσουμε το τι πραγματικά δύναται να προσφέρει το κάθε ένα.

# εγκατάσταση & σετάρισμα

## Υποδομή

Η υποδομή μας αποτελείται από ένα Ubuntu (16.04 LTS) VM, το οποίο διαχειριζόμαστε μέσω του Okeanos που μας παρέχει και τους πόρους. Το μηχάνημα αυτό διαθέτει 4 πυρήνες CPU, 8GB κύριας μνήμης, καθώς και μία δημόσια IP για πρόσβαση στο ίντερνετ.

## Σετάρισμα

Στη συνέχεια, στήσαμε την InfluxDB, ακολουθώντας τα βήματα του επίσημου οδηγού [1]. Αντίστοιχα, στήσαμε και την TimescaleDB, βασιζόμενοι στο documentation [2].

Όσον αφορά την InfluxDB σημειώνουμε ότι:

1. δεν διαθέτει δωρεάν clustered έκδοση [[1]](#endnote-2)
2. η InfluxDB 2.X δεν υποστηρίζεται από το TSBS (<https://github.com/timescale/tsbs/issues/247>, <https://github.com/timescale/tsbs/pull/209>)

ενώ όσον αφορά την TimescaleDB σημειώνουμε ότι παρόλο που διατίθεται clustered έκδοση επιλέγουμε να μην την υιοθετήσουμε για τους κάτωθι λόγους:

1. η υποστήριξη της multi-node έκδοσης έχει καταργηθεί από την έκδοση 2.13 και μετά
2. όπως έχει αποδειχθεί από τη χαμηλή υιοθέτηση της, δεν είναι κοινώς αποδεκτό ότι είναι επωφελής
3. απεναντίας, η single-node έκδοση έχει δεχθεί σημαντικές βελτιώσεις στην απόδοση εγγραφής και ανάγνωσης, (10x για συνήθη queries), ενώ μέσω αυτής βελτιστοποιείται η εκμετάλλευση της συνεισφοράς των τεχνολογιών cloud στον αναπτυξιακό τομέα.

Συμπερασματικά, αποφανθήκαμε ότι η single-node έκδοση δύναται, επί της παρούσης, να προσφέρει στην εργασία μας περισσότερο απ’ ότι η multi-node έκδοση.

## Installation

## Έπειτα, αξιοποιήσαμε την σουίτα του TSBS. Το Time Series Benchmark Suite (TSBS) είναι ένα σύνολο προγραμμάτων που έχουν γραφεί στη γλώσσα προγραμματισμού Go και χρησιμοποιούνται για να δημιουργήσουν σύνολα δεδομένων και να διεξαγάγουν δοκιμές απόδοσης, ώστε να καταγραφεί ουσιαστικά το πόσο γρήγορα μπορούν οι βάσεις μας να διαβάσουν και να γράψουν τα ζητούμενα στην εκάστοτε περίπτωση δεδομένα. Σκοπός του είναι να μπορεί να "επεκτείνεται", ώστε να βρίσκουν σε αυτό αντίκρισμα ποικιλόμορφα use cases, queries και βάσεις, με αποτέλεσμα ο κάθε ενδιαφερόμενος διαχειριστής να μπορεί να βρει την πιο κατάλληλη γι αυτόν. Τέλος, προχωρήσαμε στο installation της Go.

## ///Συνεχίσαμε σχεδιάζοντας μία δομή για τα paths, που παρουσιάζεται σε σύνοψη στα παρακάτω σχεδιάγραμματα, όπου project1 ≡ TSBS:

## Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Ολοκληρώσαμε κάνοντας run τα προαναφερόμενα συν της εντολής make, ώστε να δομηθεί το benchmark suite.

# data generation

Επόμενο βήμα μας ήταν η παραγωγή δεδομένων για αρχεία μικρού, μεσαίου και μεγάλου μεγέθους, τόσο για τη βάση InfluxDB, όσο και για την TimescaleDB (συνολικά 6 βάσεις / 6 datasets). Ενδεικτικά[[2]](#footnote-2) παρουσιάζουμε την εντολή που τρέξαμε για την βάση InfluxDB για το μεγάλο αρχείο:

* tsbs\_generate\_data--use-case="devops"--seed=123--scale=10--timestamp-start="2016-01-01T00:00:00Z"--timestamp-end="2016-02-12T00:00:00Z"--log-interval="10s"--format="influx" **>./datasets/influx\_big///**

Αποσαφηνίζουμε τα παραπάνω flags ως ακολούθως :

1. use case: σενάριο χρήσης (devops, cpu ή iot)
2. seed: κλειδί (αρχικοποίησης) για τον αλγόριθμο PRNG / εξασφαλίζει την επαναληψιμότητα της διαδικασίας δημιουργίας τυχαίων αριθμών, δεδομένου ότι με ίδιο seed προκύπτει η ίδια ακριβώς ακολουθία τυχαίων τιμών
3. scale: αριθμός συσκευών προς δημιουργία
4. timestamp-start: ώρα έναρξης
5. timestamp-end: ώρα λήξης
6. interval: πόσος χρόνος πρέπει να περάσει μεταξύ κάθε ανάγνωσης ανά συσκευή
7. format: για ποια βάση δημιουργούμε (τις InfluxDB / TimescaleDB).

Ακόμη, επί της παραγωγής των δεδομένων, σημειώνουμε ότι παρήχθησαν σε zip αρχεία, για να μην ξεπεράσουν το χώρο του υπολογιστή (30 GB), καθώς το gzip δημιούργησε περίπου υποδεκαπλάσια σε μέγεθος datasets. Επιπλέον, οι τιμές των αρχείων ορίστηκαν ως κάτωθεν, για να πληρούν τις ζητούμενες προϋποθέσεις:

* αρχείο big: minimum 8 GB (main memory)
* αρχείο medium: a few GBs (~2-3GB)
* αρχείο small: a few hundreds of MBs (~300-500 MB).

Μάλιστα, στο στάδιο αυτό παρατηρούμε το εξής:

Ενώ φτιάχναμε τα datasets για τις InfluxDB / TimescaleDB, παρόλο που είχαν τα ίδια arguments αναφορικά με τα start / end timestamps, τα intervals, αλλά και το scale, τα αρχεία είχαν σημαντικές διαφοροποιήσεις ως προς το τελικό τους μέγεθος. Ειδικότερα, η διαφορά στο μέγεθος των αρχείων οφείλεται στη χρήση διαφορετικών μορφών σειριοποίησης δεδομένων από την InfluxDB και την TimescaleDB. Όταν η συνάρτηση tsbs\_generate\_data δημιουργεί δεδομένα, τα σειριοποιεί σε μορφή κατάλληλη για τη συγκεκριμένη βάση. Στην προκειμένη, για την InfluxDB χρησιμοποιεί το line protocol της, ενώ για την TimescaleDB χρησιμοποιεί ένα format παρόμοιο με αυτό του CSV. Ωστόσο, το line protocol της InfluxDB περιλαμβάνει το όνομα της μέτρησης, το σύνολο των ετικετών, το σύνολο πεδίων και τη χρονική σήμανση για κάθε σημείο δεδομένων, ενώ τουναντίον το format της TimescaleDB, οργανώνει τα δεδομένα σε γραμμές και στήλες, με τα ονόματα των στηλών να μην επαναλαμβάνονται για κάθε σημείο των δεδομένων. Έτσι, ακόμα κι αν χρησιμοποιούνται τα ίδια δεδομένα μέτρησης, ο τρόπος σειριοποίησης κατά τη δημιουργία διαφοροποιεί τα μεγέθη των αρχείων. Συνεπώς, ουσιαστικά βλέπουμε ότι οι βάσεις της TimescaleDB έχουν έναν έξτρα πίνακα tags, ο οποίος χρησιμοποιείται ως foreign key για τα relations μεταξύ των άλλων tables. Αντιθέτως, στην InfluxDB δεν υπάρχει κάτι αντίστοιχο, οπότε όλα τα metadata αναπαράγονται σε κάθε πίνακα πολλαπλές φορές. Με αυτόν τον τρόπο, η TimescaleDB επιτυγχάνει πολύ μικρότερους χώρους στα αρχεία της, με ποσοστό περίπου 30-35%.

# DATA LOADING

Προχωρήσαμε στη φόρτωση των δεδομένων χρησιμοποιώντας συγκεκριμένα εκτελέσιμα μέσω της παρακάτω εντολής:

## cat./datasets/timescale\_medium.gz|gunzip|tsbs\_load\_timescaledb--host="localhost" --port=5432--pass="12345678"--user="postgres"--workers=2--in-table-partition-tag=true--chunk-time=8h--field-index-count=1--do-create-db=false --force-text-format=false--do-abort-on-exist=false--db-name="bigdata"

Διευκρινίζουμε ότι στην παραπάνω εντολή το μόνο flag που στην πραγματικότητα μεταβάλλουμε είναι ο αριθμός των workers, που δηλώνει την ποσότητα αυτών που χρειάζονται για να φορτώσουν τα δεδομένα. Ωστόσο, προσαρμόζουμε το host, καθώς τρέχουμε τοπικά, το user, αφού σαν χρήστης τέθηκε το postgres, το password μας (12345678) και το db name που ορίστηκε ως bigdata. Οι υπόλοιπες παραμέτροι, παραδείγματος χάριν το chunk time και το field, είναι ορισμένες από default και οι τιμές του παραμένουν σταθερές.

# query generation

Αφού κάναμε το load, προκειμένου να εξετάσουμε τα αποτελέσματα που προέκυψαν , δημιουργούμε τα queries, που κάνοντας τα run θα φανεί αν υπάρχει κάποιο πρόβλημα έως εδώ. Αυτό πραγματοποιείται με τη συνάρτηση tsbs\_generate\_queries, όπου τα flags αντιστοιχούν στα flags από το data generation που αναλύθηκαν στην αντίστοιχη παράγραφο. Σημειώνουμε ότι οι τιμές τους ταυτίζονται με αυτές κατά του data generation, με εξαίρεση το timestamp\_end που θέτεται κατά ένα δευτερόλεπτο πιο μετά, ενώ αποφασίσαμε να διατηρήσουμε σταθερό αριθμό queries (1000 σταθερά) για κάθε query type. Επίσης, τα ζιπάρουμε, για να πιάσουν λιγότερο χώρο στον δίσκο. Συμπληρωματικά, αποσαφηνίζουμε τα επιπλέον flags:

1. queries: αριθμός των queries

2. query-type: τύπος επερωτήματος

Τελικώς, τα queries θα αποθηκευτούν στο path: ///

Παρατίθεται ενδεικτικά και η εντολή για το Timescale single-group-1-1-1 query:

* tsbs\_generate\_queries--use-case="devops"--seed=123 --scale=10 --timestamp-start="2016-01-01T00:00:00Z"--timestamp-end="2016-01-02T13:45:01Z"--queries=10--query-type="single-groupby-1-1-1"--format="timescaledb"|gzip>./queries/timescaledb\_single111.gz

# Measurement of relevant performance metrics

1. *Query processing & load performance statistics*

Έχοντας φορτώσει τα δεδομένα κι αφότου δημιουργήσαμε τα queries, όπως περιεγράφηκε στην προηγούμενη ενότητα, καλούμε τη συνάρτηση tsbs\_run\_queries\_, για να τρέξουμε τα queries, να μετρήσουμε την απόδοση των ερωτημάτων αυτών και τελικώς να μπορέσουμε να προβούμε στην επιθυμητή σύγκριση των βάσεων μας. Αναφορικά με τα flags στο σημείο αυτό, αξίζει μόνο να σημειώσουμε ότι στην προκειμένη ως “workers” ορίζουμε τον αριθμό αυτών στους οποίους κατανέμεται να τρέξουν τα queries.

Παρατίθενται ενδεικτικά και οι εντολές:

* cat./queries/timescaledb\_single111.gz|gunzip|tsbs\_run\_queries\_timescaledb--workers=8--postgres="host=localhost password=12345678 user=postgres sslmode=disable database=small"
* cat./queries/influx\_single111.gz| gunzip|tsbs\_run\_queries\_influx--workers=2--db-name=small

Επίσης, αξίζει να εμβαθύνουμε στο τι εκφράζουν οι μετρικές που αξιοποιούμε για τη μέτρηση του performance. Πιο αναλυτικά, έχουμε:

1. Wall clock time: Αναφέρεται στον πραγματικό χρόνο που περνάει από την αρχή μέχρι το τέλος μιας διεργασίας, συμπεριλαμβάνοντας όλους τους χρόνους αναμονής και είναι ουσιαστικά ο χρόνος που αντιλαμβανόμαστε ως χρήστες
2. Queries/second: Ο μέσος όρος των εκτελεσμένων ερωτημάτων ανά μονάδα χρόνου, που μας δίνει μια εικόνα της ταχύτητας απόκρισης της βάσης δεδομένων
3. User time: Ο χρόνος CPU που αφιερώνεται στην εκτέλεση των εργασιών στο όνομα του χρήστη κι αντιπροσωπεύει τον χρόνο που η CPU περνά εκτελώντας τον κώδικα της εφαρμογής
4. System time: Ο χρόνος CPU που αφιερώνεται σε συστημικές εργασίες, όπως είναι οι κλήσεις συστήματος
5. CPU Usage: Το ποσοστό της CPU που χρησιμοποιείται κατά τη διάρκεια εκτέλεσης μιας διεργασίας.

Υψηλά ποσοστά CPU usage υποδηλώνουν έντονη δραστηριότητα

1. Total\_time: Ο συνολικός χρόνος που απαιτείται για την εκτέλεση μιας ερώτησης ή διαδικασίας
2. Rows/second: Ο μέσος όρος των εισαχθέντων σειρών δεδομένων από μια ερώτηση ανά δευτερόλεπτο
3. Metrics/second: Ο μέσος όρος των μετρήσεων ή μετρικών που εισάγονται στη βάση δεδομένων ανά δευτερόλεπτο.

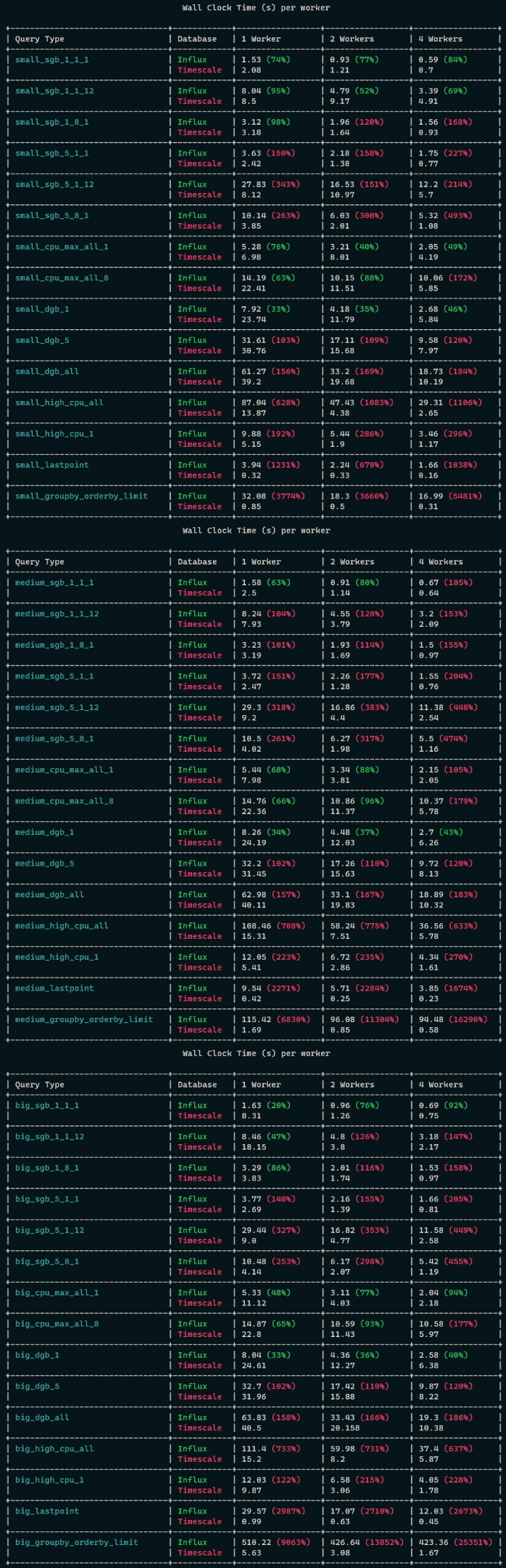
Δεδομένου, λοιπόν, ότι οι μετρικές απόδοσης παρέχουν σημαντικές πληροφορίες για τα συστήματα των βάσεων δεδομένων που μελετάμε και λαμβάνοντας υπόψη την ανάλυση που προηγήθηκε, υπογραμμίζουμε ότι σε ορισμένες μετρικές, επιδιώκουμε να έχουν χαμηλές τιμές, ενώ σε άλλες επιθυμούμε τις μεγαλύτερες δυνατές. Από όσες αναλύσαμε, το να έχουμε ως αποτέλεσμα, όσο μικρότερη τιμή τόσο το καλύτερο, εμπίπτει στις: wall clock time, user time, system time, CPU usage και total\_time, ενώ πλεονεκτούν οι μεγαλύτερες στις: queries/sec, rows/sec και metrics/sec με την λογική ότι η αύξηση του συνολικού χρόνου υποδεικνύει περισσότερες διεργασίες.

Υποσημειώνουμε τα queries που χρησιμοποιήθηκαν:

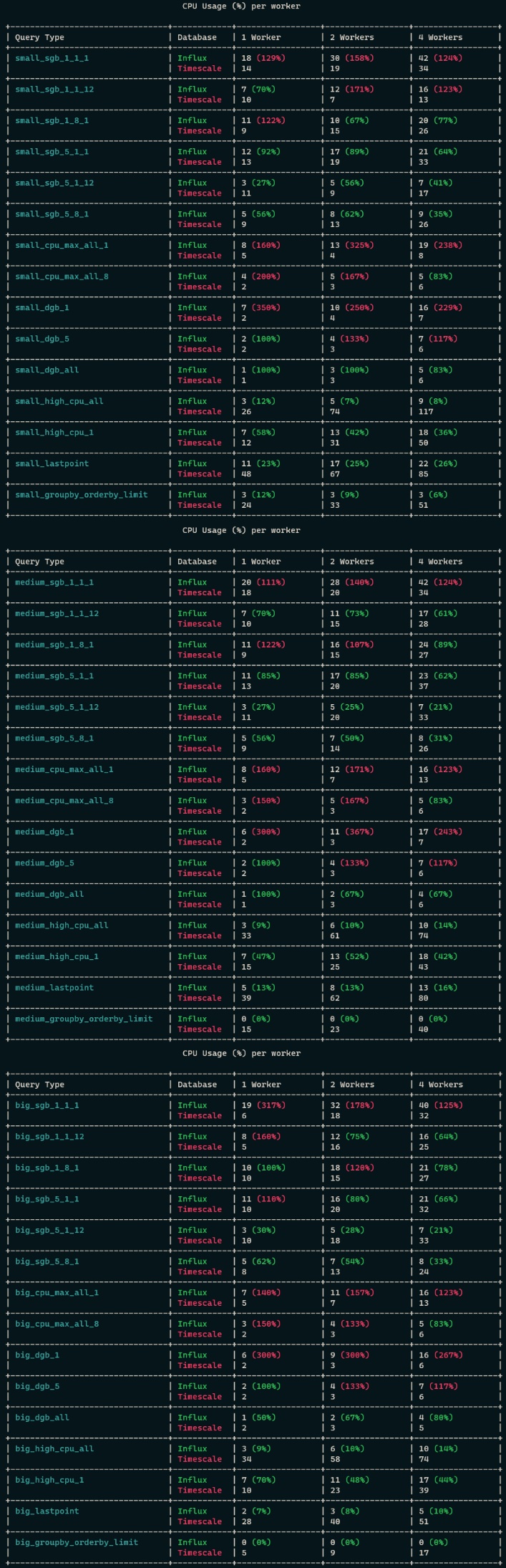


1. *Πειραματικά αποτελέσματα*

Παρατίθενται στη συνέχεια τα αποτελέσματα των μετρήσεων μας, υπό μορφής πίνακα, για κάθε μία εκ των υπό εξέταση μετρικών για τα αρχεία small, medium, big διαδοχικά, διατεταγμένα με τη σειρά που αναλύθηκαν στην προηγούμενη παράγραφο.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κατάλογος

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

1. *Παρατηρήσεις & συμπεράσματα*

Τα αποτελέσματα αντιστοιχούν σε 1000 queries ανά query type κι είναι εκφραμένα επί %. Επίσης, επισημαίνεται και η σχετική απόδοση του TimescaleDB σε σύγκριση με το InfluxDB, δηλώνοντας ουσιαστικά πόσο επί τοις εκατό των πόρων που χρησιμοποιεί το TimescaleDB χρησιμοποιεί το InfluxDB. Μάλιστα, ανάλογα με τις επιθυμητές τιμές των μετρικών, η ποσοστιαία αυτή σχέση σημειώνεται με το χρώμα του «νικητή»· κόκκινο για όταν υπερτερεί το TimescaleDB έναντι του InfluxDB και πράσινο για το αντίστροφο. Προς διευκόλυνσή μας, κατηγοριοποιούμε τα queries ως εξής:

* Simple rollups: single-groupby-1-1-1/ -1-1-12/ -1-8-1/ -5-1-1/ -5-1-12/ -5-8-1
* Aggregates: cpu-max-all-1/ -all-8
* Double rollups: double-groupby-1/ -5/ -all
* Thresholds: high-cpu-all/ -1
* Complex queries: lastpoint, groupby-orderby-limit

Σε πρώτο στάδιο, πραγματοποιούμε σύγκριση μεταξύ των μετρικών ανά κατηγορία query ανά μέγεθος αρχείου. Συνεπώς:

1. *Wall clock time*
   1. Single rollups: Παρατηρούμε ότι ανεξαρτήτως του μεγέθους του αρχείου υπερτερεί το TimescaleDB. Σημειώνουμε απλώς, ότι υπερτερεί περισσότερο στο medium αρχείο, έπειτα στο big και λιγότερο στο small.
   2. Aggregates: Υπερτερεί το InfluxDB
   3. Double rollups: Ενώ στο dgb\_1 υπερτερεί το Influx, στα άλλα δύο υπερτερεί το TimescaleDB, άρα συνολικά υπερτερεί το TimescaleDB
   4. Thresholds & Complex queries: Ομόφωνα υπερτερεί το TimescaleDB
2. *Queries/second*

Πανομοιότυπη συμπεριφορά με το wall clock time.

1. *User time* 
   1. Single rollups: Με εξαίρεση το sgb\_1\_1\_12, ανεξαρτήτως μεγέθους, υπερτερεί το TimescaleDB.
   2. Aggregates & Double rollups: Εμφανώς καλύτερη η απόδοση του TimescaleDB
   3. Thresholds: Η απόδοση του InfluxDB αντιστοιχεί στην απόδοση του TimescaleDB, εννοώντας ότι δεν υπερέχει κάποιο έναντι του άλλου
   4. Complex queries: Oλικώς υπερτερεί το TimescaleDB*.*
2. *System time*

Με εξαίρεση τα complex queries, σε όλες τις υπόλοιπες κατηγορίες, το InfluxDB ξεπερνάει σε απόδοση το TimescaleDB.

1. *CPU Usage*
   1. Single rollups: Με εξαίρεση το sgb\_1\_1, το InfluxDB πλεονεκτεί
   2. Aggregates: Ανεξαρτήτως μεγέθους αρχείου, το Timescale DB υπερτερεί
   3. Double rollups: Οι δύο βάσεις έχουν αντίστοιχες αποδόσεις
   4. Thresholds & Complex queries: Ξεκάθαρη υπεροχή του InfluxDB.

Σε δεύτερο στάδιο, παρατηρούμε ότι στο InfluxDB η συμμετοχή επιπλέον workers δεν επιφέρει σημαντική βελτίωση (το αντίθετο μάλιστα σε κάποιες περιπτώσεις), ενώ στο TimescaleDB η απόδοση βελτιώνεται πολύ καταμερίζοντας σε περισσότερους workers τα queries (ενδεικτικά οι χρόνοι υποδιπλασιάζονται ανά διπλασιασμό workers), δείχνοντας μας ότι η μεταβολή των workers έχει μεγαλύτερη επίδραση στην TimescaleDB.

1. *Total time, Rows/second & Metrics/second*

*Σε αντίθεση με όλες τις παραπάνω μετρικές για τις οποίες έγινε σύγκριση, όπου κατά γενική ομολογία, δεν παρατηρήθηκαν συμπεριφορικές διαφοροποιήσεις ανά το μέγεθος του αρχείου, στην προκειμένη, παρατηρούνται έντονες μεταβολές- που υποδηλώνει ότι στο φόρτωμα των αρχείων στην TimescaleBD και InfluxDB, καταναλώνονται σημαντικά διαφορετικοί πόροι.*

* 1. Για το small αρχείο, οι δύο βάσεις σε γενικές γραμμές φαίνεται να έχουν αντίστοιχες αποδόσεις. Ειδικότερα το InfluxDB πλεονεκτεί για τους 1 & 2 workers, ενώ το TimescaleBD υπερτερεί στους 4 & 8 workers
  2. Για το medium αρχείο, σημειώνουμε ότι δεν παρατηρείται κάποιο μοτίβο αντίστοιχο του παραπάνω ανά τους workers, ενώ υπερτερεί το TimescaleBD
  3. Για το big αρχείο, κατ΄ αύξηση των workers, υπερτερεί σημαντικά το InfluxDB.

# Σύνοψη

Προφανώς, για να αποφασίζουμε ποια βάση δεδομένων θα επιλέγουμε σε κάθε μας project, πρέπει να λαμβάνουμε υπόψη πολυποίκιλους παράγοντες, όπως είναι παραδείγματος χάριν το μοντέλο των δεδομένων, τη γλώσσα των queries, την αξιοπιστία, την απόδοση κ.α.. Στη παρούσα εργασία καταφέραμε να δημιουργήσουμε δύο λειτουργικά συστήματα βάσεων δεδομένων, να τα συγκρίνουμε εστιάζοντας στο κομμάτι της απόδοσης (κυρίως latency metrics) και αποφανθήκαμε σε ποια περίπτωση υπερτερεί ποια βάση. Αποκομίσαμε, λοιπόν, την εμπειρία της δημιουργίας και της σύγκρισης, μέσω ανάλυσης των δεδομένων μας και κριτικής σκέψης.

Ο κώδικας που χρησιμοποιήθηκε καθώς και τα scripts για τα configurations των επιμέρους εργαλείων βρίσκονται στον παρακάτω σύνδεσμο github:

<https://github.com/LavredisG/NTUA-Analysis-and-Design-of-Information-Systems>

# REFERENCES

1. [Timescale Documentation | Install TimescaleDB on Linux](https://docs.timescale.com/self-hosted/latest/install/installation-linux/)
2. [Install InfluxDB OSS | InfluxDB OSS v1 Documentation (influxdata.com)](https://docs.influxdata.com/influxdb/v1/introduction/install/)
3. [TimescaleDB vs. InfluxDB: Purpose-built for time-series data](https://www.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/)
4. <https://www.influxdata.com/comparison/influxdb-vs-timescaledb/>
5. <https://github.com/timescale/tsbs>

1. [↑](#endnote-ref-2)
2. *Το σύνολο των εντολών θα φορτωθεί για διευκόλυνση σε bash script, για να εκτελεστεί.* [↑](#footnote-ref-2)