ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κατανεμημένα Συστήματα

Εξαμηνιαία Εργασία - BlockChat

Αναστάσιος Στέφανος Αναγνώστου
03119051

24 Μαρτίου 2024

Περιεχόμενα

Μέρος I

Σχεδιασμός Συστήματος

1 Δεν είμαι σίγουρος

Μέρος II
Πειράματα

Ανά πείραμα αξιολογούνται, αφενός τα πιο χρονοβόρα κομμάτια του κώδικα, όπως υποδεικνύει το profiling καθενός κόμβου κατά την εκτέλεση του πειράματος, αφετέρου οι συναρτήσεις mint, validateTransaction και processTXs, οι οποίες συνιστούν την λογική λειτουργίας των κόμβων του συστήματος. Επίσης, εκτιμάται η ρυθμαπόδοση του συστήματος και το μέσο block time.

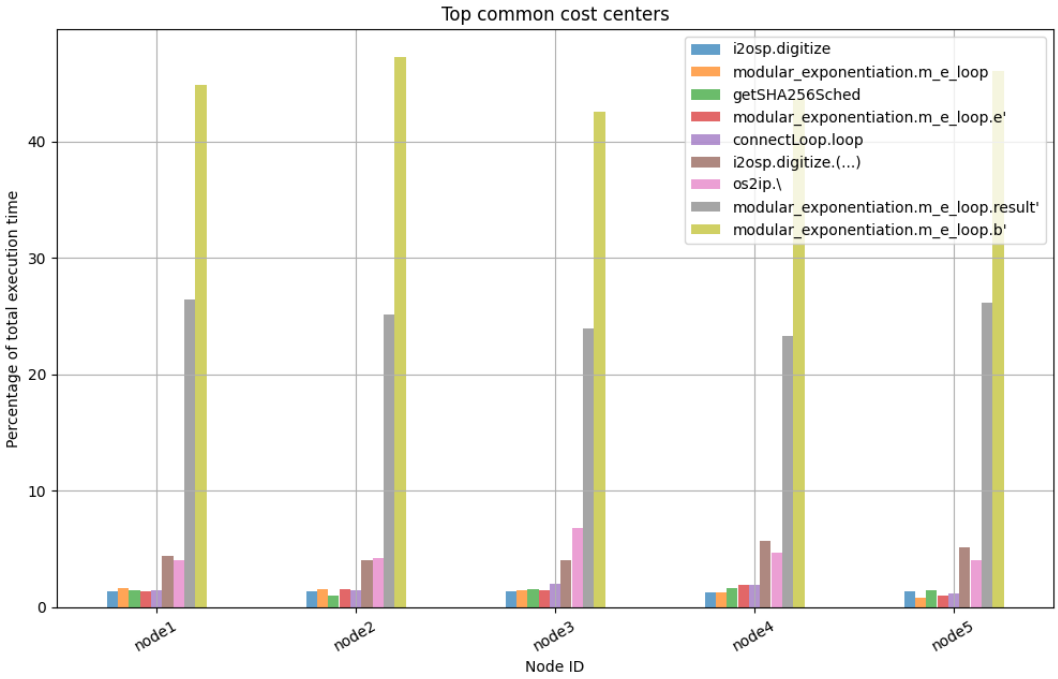
2 Απόδοση του συστήματος

Σημειώνεται ότι το ποσοστό του χρόνου εκτέλεσης των σημείων που υποδεικνύει το profiling δεν είναι κληρονομημένο, δηλαδή δεν εμπεριέχονται στο ποσοστό οι χρόνοι εκτέλεσης των συναρτήσεων που καλούνται από τις συναρτήσεις που εμφανίζονται στο profiling.

2.1 Χρονοβόρα τμήματα του κώδικα

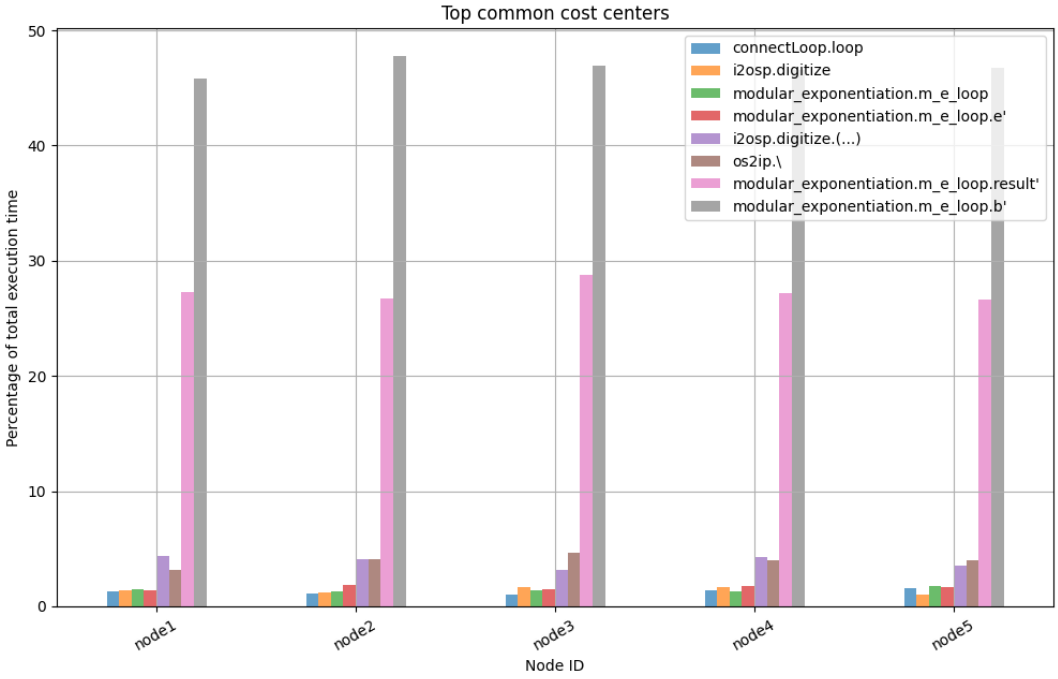
Στο πείραμα για την αξιολόγηση της ρυθμαπόδοσης του συστήματος, στήνεται ένα δίκτυο 5 κόμβων, καθένας εκ των οποίων εκτελεί 1 staking, με stake 10 BCC συναλλαγή και 50 συναλλαγές (συγκεκριμένα αποστολές μηνυμάτων) προς τους άλλους κόμβους. Η ταχύτητα αποστολής συναλλαγών είναι ίδια μεταξύ των κόμβων, ίση με $2\frac{txs}{s}$ και παραμένει σταθερή μεταξύ όλων των πειραμάτων.

Το πρώτο πράγμα που φαίνεται στο σχήμα 1 είναι ότι το μακράν πιο χρονοβόρο μέρος του κώδικα είναι η συνάρτηση modular_exponentiation που χρησιμοποιείται γενικά για την κρυπτογράφηση / αποκρυπτογράφηση και υπογραφή / επαλήθευση μηνυμάτων. Συγκεκριμένα, φαίνεται να λαμβάνει περίπου το 45% του συνολικού χρόνου υπολογισμού[1] του προγράμματος.

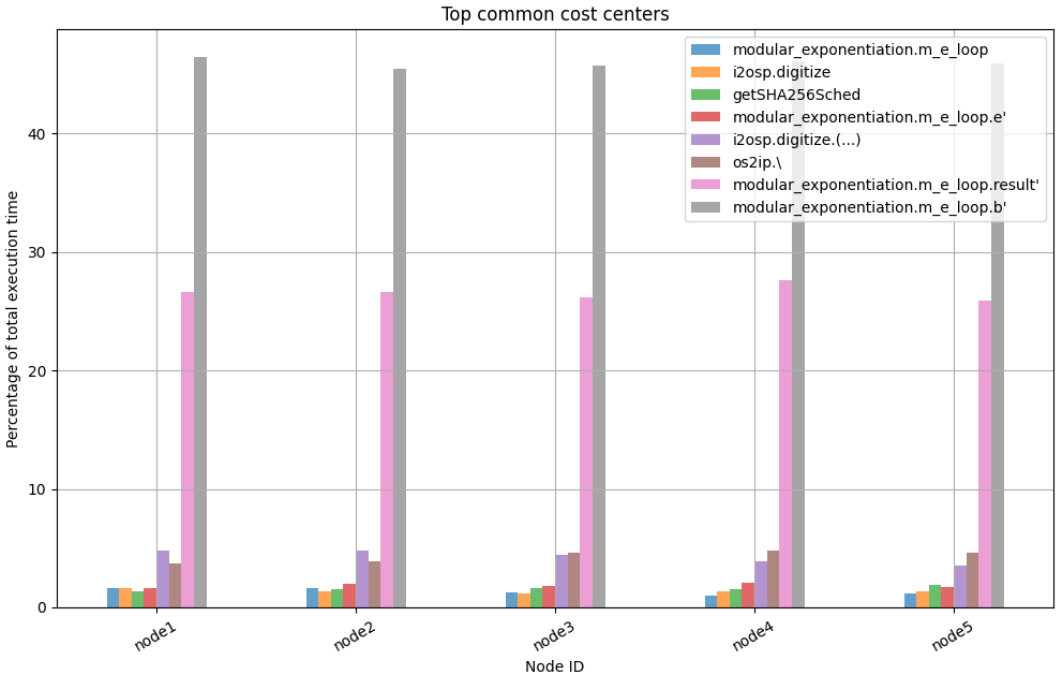


(α′) capacity=5

[1] Ο profiler της Haskell μετράει *CPU time* όχι *blocking time*



(β′) capacity=10

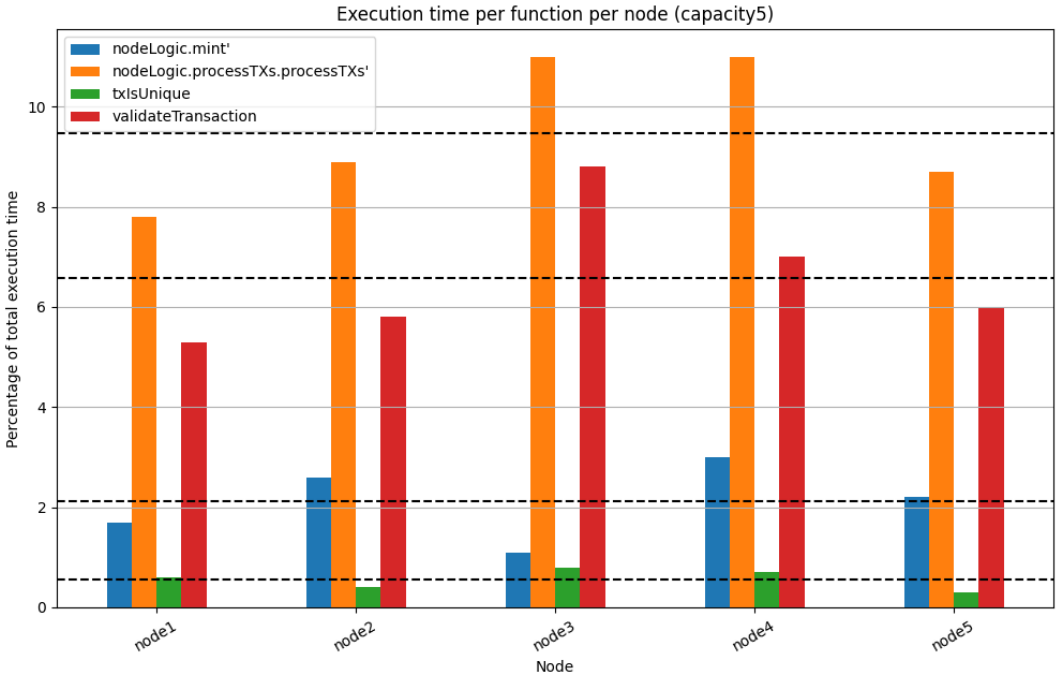


(γ′) capacity=20

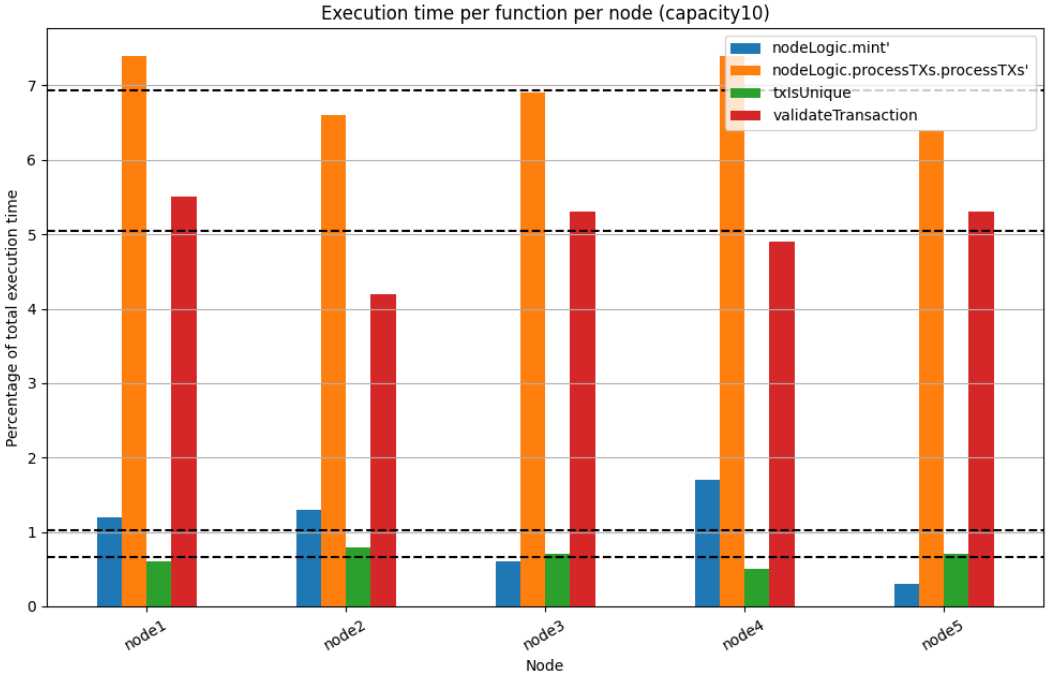
Σχήμα 1: Τα πιο χρονοβόρα κομμάτια του κώδικα

2.2 Συναρτήσεις του συστήματος

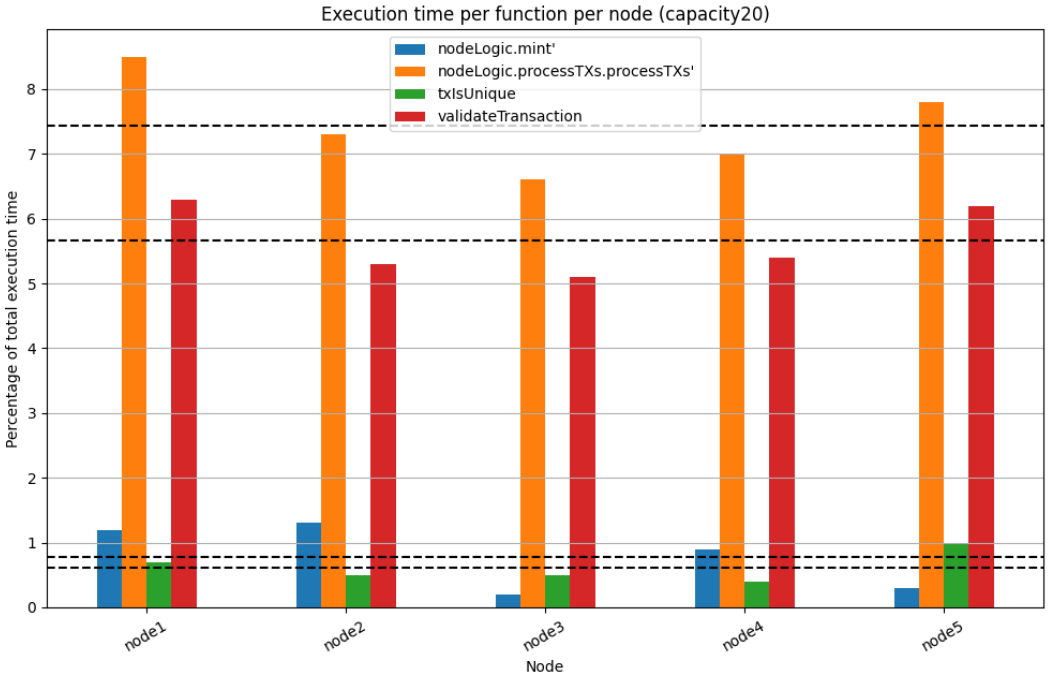
Σχετικά με τις top level συναρτήσεις του συστήματος, παρατηρείται ότι, με κάποιες μικρές διακυμάνσεις, η processTXs καταναλώνει 9-10% του συνολικού CPU time, η mint 1-3% και η validateTransaction 5-9%, με μέσο όρο περίπου 6.5%.



(α΄) Ρυθμαπόδοση capacity=5



(β′) Ρυθμαπόδοση capacity=10



(γ′) Ρυθμαπόδοση capacity=20

Σχήμα 2: Ποσοστό χρόνου επί του συνολικού χρόνου εκτέλεσης που λαμβάνει η κάθε συνάρτηση

Στον πίνακα 1 παρουσιάζονται ορισμένα στατιστικά σχετικά με τις συναρτήσεις processTXs, validateTransaction, txIsUnique και mint. Το πιο σημαντικό να παρατηρηθεί είναι ότι, για κάθε κόμβο, οι κλήσεις στην συνάρτηση validateTransaction είναι ακριβώς τόσες όσες και οι συναλλαγές που αποστέλλονται από όλους τους κόμβους ($5 + 5 \times 50 = 255$). Επίσης, $\#mint + \#validateTransaction = \#processTXs$ [2]. Παρ΄ότι φαίνεται σαν να επικυρώθηκαν όλες οι συναλλαγές, αυτό δεν ισχύει. Στην πραγματικότητα, επειδή οι κόμβοι δεν παραλαμβάνουν κατ΄ ανάγκην τις συναλλαγές με την σειρά αποστολή τους, είναι πιθανό κάποιος validator να ακυρώσει κάποια συναλλαγή η οποία με διαφορετική σειρά θα είχε επιβεβαιωθεί. Για αυτόν τον λόγο φαίνεται ότι ένα υποσύνολο των συναλλαγών εξετάζεται για την μοναδικότητά τους από την συνάρτηση txIsUnique. Έτσι αιτιολογείται και το γεγονός ότι το blockchain έχει μήκος μικρότερο από το μέγιστο δυνατό του δεδομένων των συναλλαγών. Παραδείγματος χάριν, για capacity 5 έχει μήκος $41 = \frac{207}{5} < \frac{255}{5} = 51$.

Πίνακας 1: Στατιστικά συναρτήσεων ανά κόμβο

(α΄) capacity=5

Node	**Function**	**Entries**	**TimeInh**
node1.prof:	nodeLogic.processTXs.processTXs'	264	7.8
node1.prof:	validateTransaction	223	5.3
node1.prof:	txIsUnique	207	0.6
node1.prof:	nodeLogic.mint'	40	1.7
node2.prof:	nodeLogic.processTXs.processTXs'	265	8.9
node2.prof:	validateTransaction	223	5.8
node2.prof:	txIsUnique	207	0.4
node2.prof:	nodeLogic.mint'	41	2.6
node3.prof:	nodeLogic.processTXs.processTXs'	263	11.0
node3.prof:	validateTransaction	223	8.8
node3.prof:	txIsUnique	201	0.8
node3.prof:	nodeLogic.mint'	39	1.1
node4.prof:	nodeLogic.processTXs.processTXs'	263	11.0
node4.prof:	validateTransaction	223	7.0
node4.prof:	txIsUnique	201	0.7
node4.prof:	nodeLogic.mint'	39	3.0
node5.prof:	nodeLogic.processTXs.processTXs'	265	8.7
node5.prof:	validateTransaction	223	6.0
node5.prof:	txIsUnique	209	0.3
node5.prof:	nodeLogic.mint'	41	2.2

Όπως φαίνεται από τον πίνακα, λοιπόν, για τον υπολογισμό του block time και της ρυθμαπόδοσης λαμβάνονται υπόψιν τόσα μπλοκς όσα και οι κλήσεις στην συνάρτηση mint και τόσες συναλλαγές όσες τα μπλοκς επί την εκάστοτε χωρητικότητα.

$$\begin{aligned} &\text{Χωρητικότητα} = 5 \Rightarrow \text{Μπλοκς} = 41 \text{ και } \Sigma\text{υναλλαγές} = 205 \\ &\text{Χωρητικότητα} = 10 \Rightarrow \text{Μπλοκς} = 22 \text{ και } \Sigma\text{υναλλαγές} = 220 \\ &\text{Χωρητικότητα} = 20 \Rightarrow \text{Μπλοκς} = 11 \text{ και } \Sigma\text{υναλλαγές} = 220 \end{aligned} \tag{1}$$

[2] Η -1 διαφορά είναι επειδή έγινε η τελευταία κλήση και τα προγράμματα έλαβαν σήμα τερματισμού

	(β′) capacity=10			(γ′) capacity=20	
Node	**Function**	**Entries**	**TimeInh**	**Entries**	**TimeInh**
node1.prof:	nodeLogic.processTXs.processTXs’	278	7.4	267	8.5
node1.prof:	validateTransaction	255	5.5	255	6.3
node1.prof:	txIsUnique	228	0.6	229	0.7
node1.prof:	nodeLogic.mint’	22	1.2	11	1.2
node2.prof:	nodeLogic.processTXs.processTXs’	278	6.6	267	7.3
node2.prof:	validateTransaction	255	4.2	255	5.3
node2.prof:	txIsUnique	229	0.8	229	0.5
node2.prof:	nodeLogic.mint’	22	1.3	11	1.3
node3.prof:	nodeLogic.processTXs.processTXs’	278	6.9	267	6.6
node3.prof:	validateTransaction	255	5.3	255	5.1
node3.prof:	txIsUnique	227	0.7	223	0.5
node3.prof:	nodeLogic.mint’	22	0.6	11	0.2
node4.prof:	nodeLogic.processTXs.processTXs’	278	7.4	267	7.0
node4.prof:	validateTransaction	255	4.9	255	5.4
node4.prof:	txIsUnique	226	0.5	227	0.4
node4.prof:	nodeLogic.mint’	22	1.7	11	0.9
node5.prof:	nodeLogic.processTXs.processTXs’	278	6.4	267	7.8
node5.prof:	validateTransaction	255	5.3	255	6.2
node5.prof:	txIsUnique	227	0.7	229	1.0
node5.prof:	nodeLogic.mint’	22	0.3	11	0.3

2.3 Ρυθμαπόδοση και Block time

Το block time μπορεί να υπολογιστεί λαμβάνοντας τον μέσο όρο των διαφορών των time stamps διαδοχικών blocks. Στο σχήμα 3 φαίνονται οι χρόνοι δημιουργίας block όπως υπολογίστηκαν από κάθε κόμβο. Παρατηρείται ότι δεν είναι πάντοτε ίσοι μεταξύ των κόμβων. Αυτό συμβαίνει γιατί, κατά τον τερματισμό του πειράματος, δεν έχουν φτάσει κατάνάγκην όλοι οι κόμβοι στο ίδιο σημείο της αλυσίδας και για αυτόν τον λόγο διαφοροποιείται η μέτρησή τους. Εδώ λαμβάνεται υπόψιν το μέγιστο μήκος της αλυσίδας μεταξύ των κόμβων.

```
capacity5/node1.log:Mean time between blocks: 625.0
capacity5/node2.log:Mean time between blocks: 658.5853658536586
capacity5/node3.log:Mean time between blocks: 641.025641025641
capacity5/node4.log:Mean time between blocks: 641.025641025641
capacity5/node5.log:Mean time between blocks: 658.5853658536586

capacity10/node1.log:Mean time between blocks: 1045.5
capacity10/node2.log:Mean time between blocks: 1045.5
capacity10/node3.log:Mean time between blocks: 1045.5
capacity10/node4.log:Mean time between blocks: 1045.5
capacity10/node5.log:Mean time between blocks: 1045.5

capacity20/node1.log:Mean time between blocks: 2000.3636363636363
capacity20/node2.log:Mean time between blocks: 2000.3636363636363
capacity20/node3.log:Mean time between blocks: 2000.3636363636363
capacity20/node4.log:Mean time between blocks: 2000.3636363636363
capacity20/node5.log:Mean time between blocks: 2000.3636363636363
```

Σχήμα 3: Μέσος χρόνος δημιουργίας block (ms)

Από τον χρόνο αυτόν μπορούν να μετρηθούν και οι εξυπηρετούμενες συναλλαγές ανά δευτερόλεπτο. Συγκεκριμένα, μία συναλλαγή εξυπηρετείται όταν επικυρωθεί, δηλαδή όταν καταγραφεί στην αλυσίδα. Άρα, για κάθε capacity έχουμε:

$$\text{Ρυθμαπόδοση} = \frac{\text{Συναλλαγές}}{\text{Χρόνος}} = \frac{\text{Συναλλαγές}}{\text{Μπλοκ}} \cdot \frac{\text{Μπλοκ}}{\text{Χρόνος}} \Leftrightarrow$$

$$\text{Ρυθμαπόδοση} = \frac{\frac{\text{Συναλλαγές}}{\text{Μπλοκ}}}{\text{Μέσος χρόνος δημιουργίας μπλοκ}} = \frac{\text{Χωρητικότητα}}{\text{Μέσος χρόνος δημιουργίας μπλοκ}} \Rightarrow$$

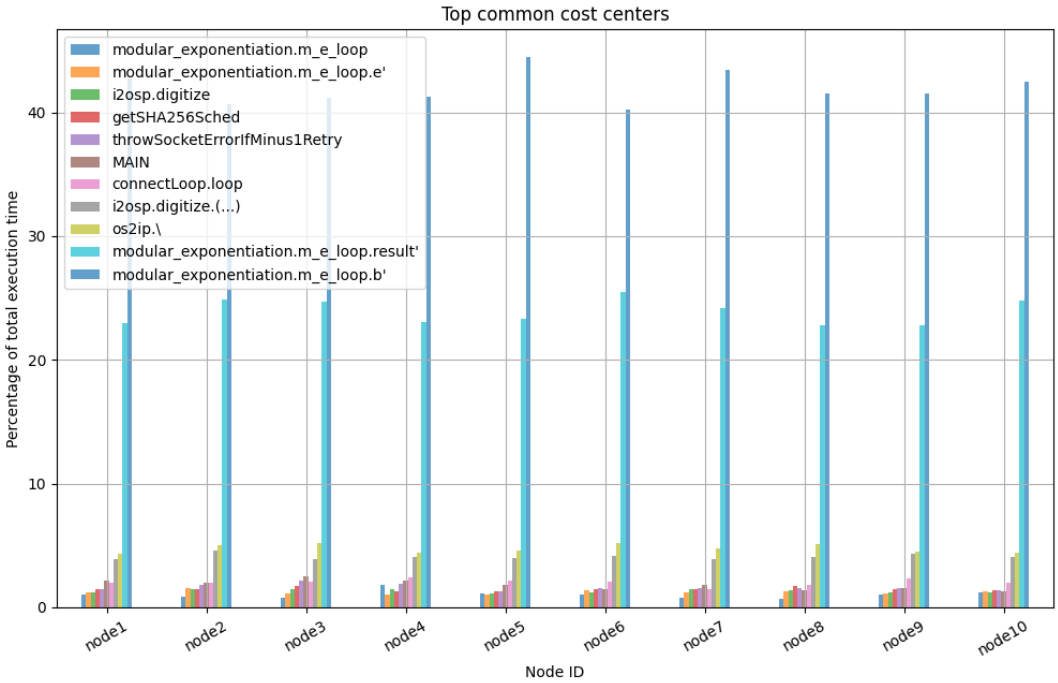
$$\boxed{\begin{aligned} \text{Ρυθμαπόδοση}_{capacity=5} &= \frac{5}{0,658} = 7,598\frac{txs}{s} \\ \text{Ρυθμαπόδοση}_{capacity=10} &= \frac{10}{1,045} = 9,569\frac{txs}{s} \\ \text{Ρυθμαπόδοση}_{capacity=20} &= \frac{20}{2} = 10\frac{txs}{s} \end{aligned}} \tag{2}$$

3 Κλιμακωσιμότητα του συστήματος

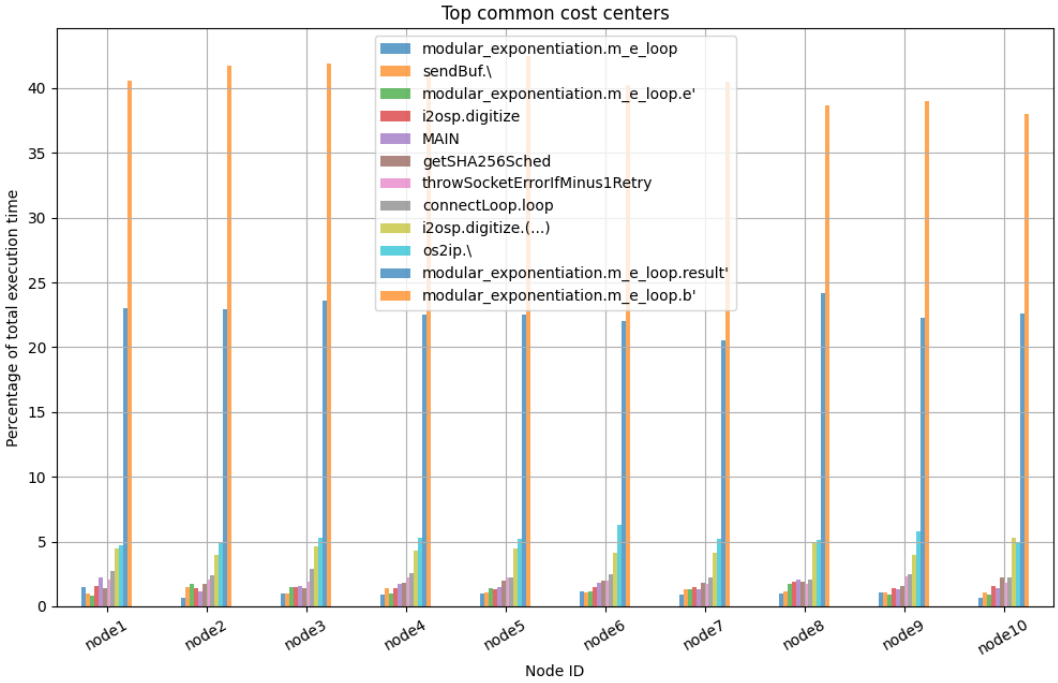
Στο πείραμα κλιμακωσιμότητας, το δίκτυο εκκινείται με 10 κόμβους, καθένας εκ των οποίων εκτελεί 1 staking συναλλαγή, με stake 10 BCC και 100 συναλλαγές (συγκεκριμένα αποστολές μηνυμάτων) προς τους άλλους κόμβους. Σκοπός είναι να εξεταστεί η κλιμάκωση του συστήματος ως προς το πλήθος των συμμετεχόντων κόμβων.

3.1 Χρονοβόρα τμήματα του κώδικα

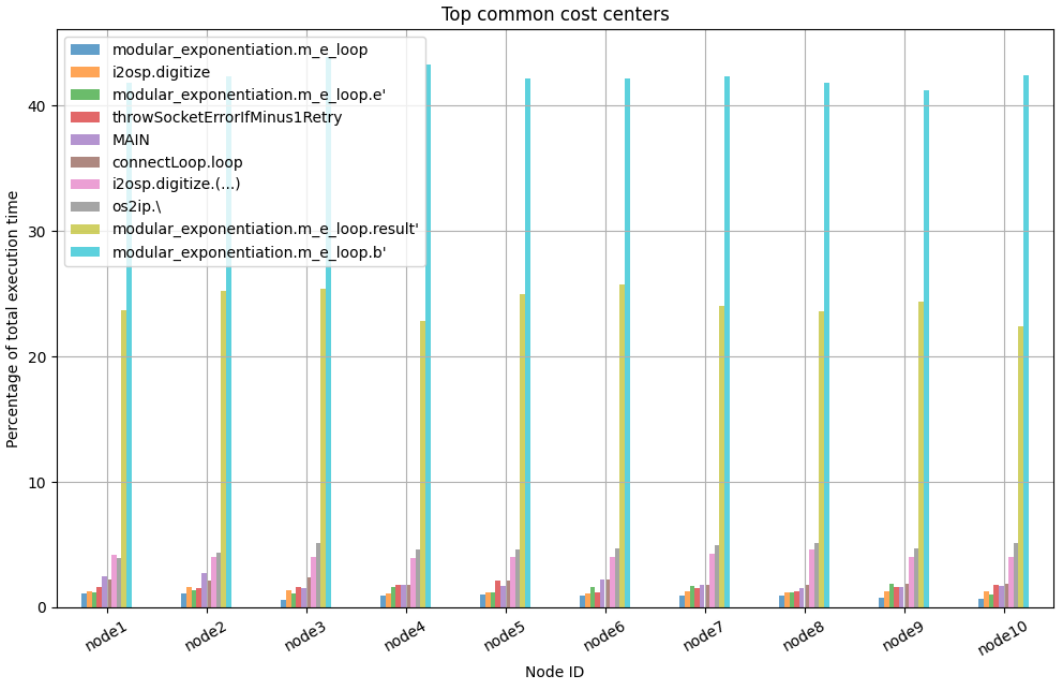
Στα γραφήματα 4 φαίνονται τα πιο χρονοβόρα κομμάτια του κώδικα για κάθε πείραμα κλιμακωσιμότητας. Φαίνεται ότι αυτά είναι τα ίδια με τα πιο χρονοβόρα κομμάτια του κώδικα για το πείραμα ρυθμαπόδοσης, με την συνάρτηση modular_exponentiation να καταλαμβάνει πάλι περίπου το 45% του συνολικού CPU time του προγράμματος.



(α′) capacity=5



(β′) capacity=10

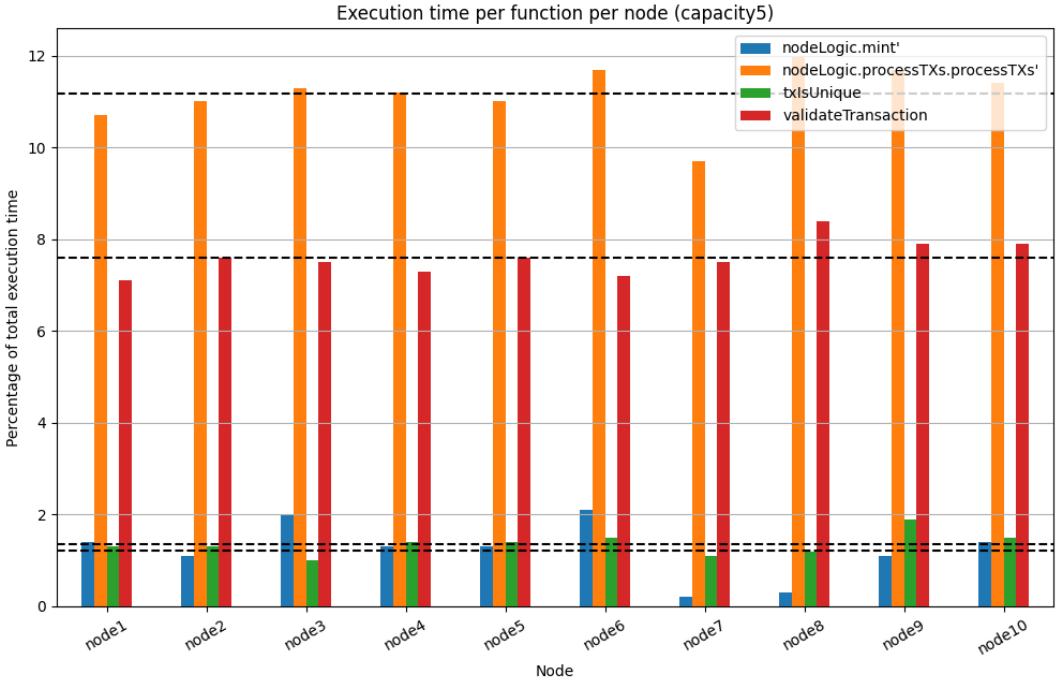


(γ′) capacity=20

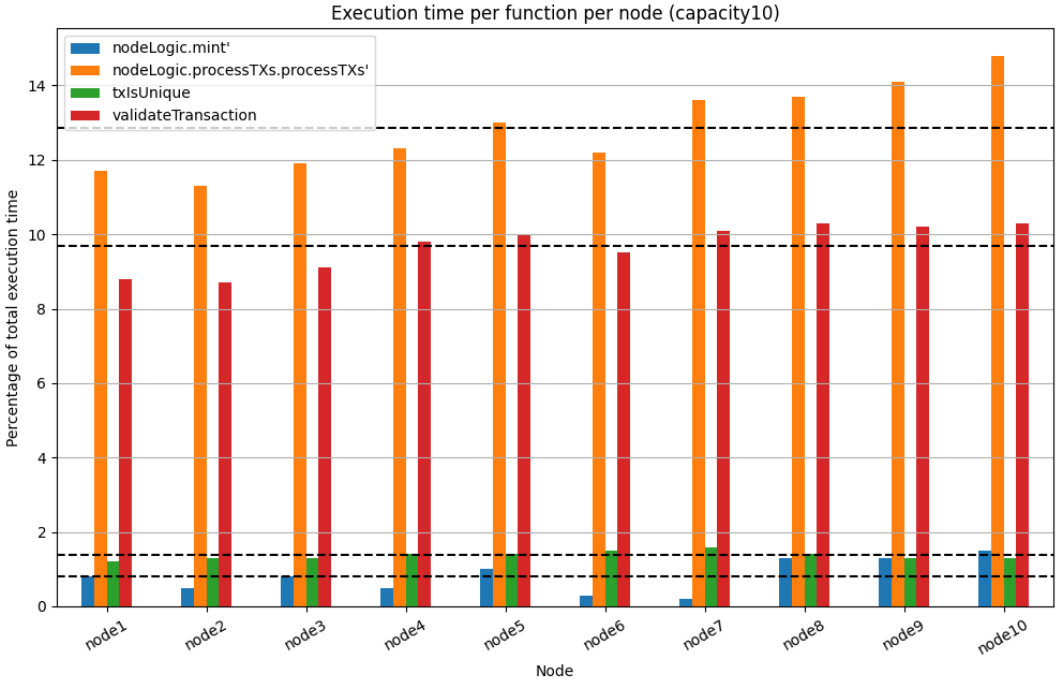
Σχήμα 4: Τα πιο χρονοβόρα κομμάτια του κώδικα

3.2 Συναρτήσεις του συστήματος

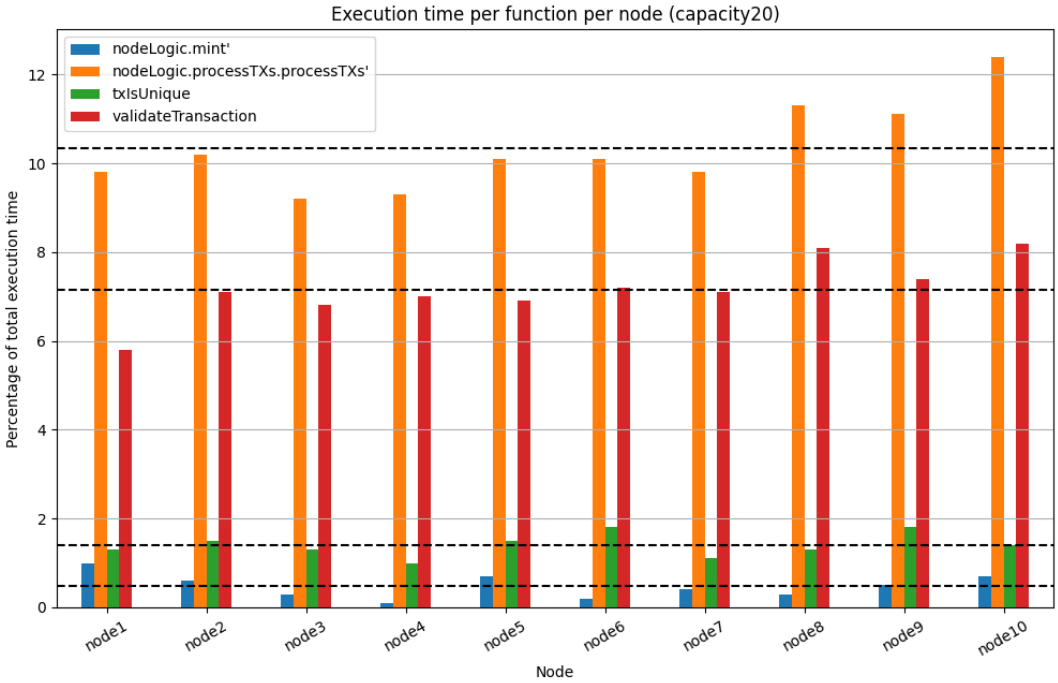
Στα γραφήματα 5 παρατηρείται ότι οι συναρτήσεις καταλαμβάνουν περίπου το ίδιο ποσοστό χρόνου εκτέλεσης με το προηγούμενο πείραμα.



(α′) capacity=5



(β′) capacity=10



(γ′) capacity=20

Σχήμα 5: Ποσοστό χρόνου επί του συνολικού χρόνου εκτέλεσης που λαμβάνει η κάθε συνάρτηση

Στον πίνακα 2α′ φαίνονται οι κλήσεις ενδιαφέροντος των κόμβων. Παρατηρώντας το πλήθος των κλήσεων ανά συνάρτηση, διαπιστώνεται ότι οι κόμβοι δεν προλαβαίνουν να επικυρώσουν όλες τις συναλλαγές που λαμβάνουν. Αυτό οφείλεται, αφενός στον μεγαλύτερο όγκο συναλλαγών $10 + 10 \times 100 = 1010$ ο οποίος είναι ≈ 4 φορές μεγαλύτερος από προηγουμένως, και αφετέρου στην μικρή χωρητικότητα του block, το οποίο σημαίνει ότι οι κόμβοι πρέπει συχνά να καλούν την χρονοβόρα συνάρτηση mint και να διακόπτουν την διαδικασία επικύρωσης.

Επίσης, παρατηρείται ότι οι κόμβοι 7-8 έχουν μείνει πολύ πίσω σε σχέση με τους υπόλοιπους κόμβους. Αυτό είναι μάλλον συνέπεια της πειραματικής διάταξης, αφού όλοι οι κόμβοι τρέχουν στο ίδιο μηχάνημα.

Πίνακας 2: Στατιστικά συναρτήσεων ανά κόμβο

(α′) capacity=5

Node	**Function**	**Entries**	**TimeInh**
node10.prof:	nodeLogic.processTXs.processTXs’	1111	11.4
node10.prof:	validateTransaction	1010	7.9
node10.prof:	txIsUnique	545	1.5
node10.prof:	nodeLogic.mint’	100	1.4
node1.prof:	nodeLogic.processTXs.processTXs’	1111	10.7
node1.prof:	validateTransaction	1010	7.1
node1.prof:	txIsUnique	544	1.3
node1.prof:	nodeLogic.mint’	100	1.4
node2.prof:	nodeLogic.processTXs.processTXs’	1111	11.0
node2.prof:	validateTransaction	1010	7.6
node2.prof:	txIsUnique	542	1.3
node2.prof:	nodeLogic.mint’	100	1.1
node3.prof:	nodeLogic.processTXs.processTXs’	1111	11.3
node3.prof:	validateTransaction	1010	7.5
node3.prof:	txIsUnique	544	1.0
node3.prof:	nodeLogic.mint’	100	2.0
node4.prof:	nodeLogic.processTXs.processTXs’	1111	11.2
node4.prof:	validateTransaction	1010	7.3
node4.prof:	txIsUnique	544	1.4
node4.prof:	nodeLogic.mint’	100	1.3
node5.prof:	nodeLogic.processTXs.processTXs’	1111	11.0
node5.prof:	validateTransaction	1010	7.6
node5.prof:	txIsUnique	543	1.4
node5.prof:	nodeLogic.mint’	100	1.3
node6.prof:	nodeLogic.processTXs.processTXs’	1111	11.7
node6.prof:	validateTransaction	1010	7.2
node6.prof:	txIsUnique	542	1.5
node6.prof:	nodeLogic.mint’	100	2.1
node7.prof:	nodeLogic.processTXs.processTXs’	1101	9.7
node7.prof:	validateTransaction	1010	7.5
node7.prof:	txIsUnique	495	1.1
node7.prof:	nodeLogic.mint’	90	0.2
node8.prof:	nodeLogic.processTXs.processTXs’	1101	12.0
node8.prof:	validateTransaction	1010	8.4
node8.prof:	txIsUnique	496	1.2
node8.prof:	nodeLogic.mint’	90	0.3
node9.prof:	nodeLogic.processTXs.processTXs’	1111	11.7
node9.prof:	validateTransaction	1010	7.9
node9.prof:	txIsUnique	546	1.9
node9.prof:	nodeLogic.mint’	100	1.1

Αντιθέτως, στους πίνακες 2β′ και 2γ′ φαίνεται από τις κλήσεις των συναρτήσεων ότι έχουν επικυρωθεί όλες οι συναλλαγές και έχουν παραχθεί τα αντίστοιχα blocks. Η μεγαλύτερη χωρητικότητα των blocks επιτρέπει στους κόμβους μεγαλύτερα χρονικά παράθυρα για την επικύρωση των συναλλαγών και η διακοπή για την παραγωγή των blocks δεν καθυστερεί την εξέλιξη του δικτύου.

	(β′) capacity=10			(γ′) capacity=20	
Node	**Function**	**Entries**	**TimeInh**	**Entries**	**TimeInh**
node10.prof:	nodeLogic.processTXs.processTXs’	1061	14.8	1024	12.4
node10.prof:	validateTransaction	1010	10.3	999	8.2
node10.prof:	txIsUnique	503	1.3	501	1.4
node10.prof:	nodeLogic.mint’	50	1.5	25	0.7
node1.prof:	nodeLogic.processTXs.processTXs’	1061	11.7	986	9.8
node1.prof:	validateTransaction	1010	8.8	961	5.8
node1.prof:	txIsUnique	503	1.2	500	1.3
node1.prof:	nodeLogic.mint’	50	0.8	25	1.0
node2.prof:	nodeLogic.processTXs.processTXs’	1061	11.3	992	10.2
node2.prof:	validateTransaction	1010	8.7	967	7.1
node2.prof:	txIsUnique	504	1.3	500	1.5
node2.prof:	nodeLogic.mint’	50	0.5	25	0.6
node3.prof:	nodeLogic.processTXs.processTXs’	1061	11.9	968	9.2
node3.prof:	validateTransaction	1010	9.1	943	6.8
node3.prof:	txIsUnique	505	1.3	500	1.3
node3.prof:	nodeLogic.mint’	50	0.8	25	0.3
node4.prof:	nodeLogic.processTXs.processTXs’	1061	12.3	1033	9.3
node4.prof:	validateTransaction	1010	9.8	1010	7.0
node4.prof:	txIsUnique	506	1.4	454	1.0
node4.prof:	nodeLogic.mint’	50	0.5	22	0.1
node5.prof:	nodeLogic.processTXs.processTXs’	1061	13.0	990	10.1
node5.prof:	validateTransaction	1010	10.0	965	6.9
node5.prof:	txIsUnique	505	1.4	500	1.5
node5.prof:	nodeLogic.mint’	50	1.0	25	0.7
node6.prof:	nodeLogic.processTXs.processTXs’	1061	12.2	1035	10.1
node6.prof:	validateTransaction	1010	9.5	1010	7.2
node6.prof:	txIsUnique	504	1.5	497	1.8
node6.prof:	nodeLogic.mint’	50	0.3	24	0.2
node7.prof:	nodeLogic.processTXs.processTXs’	1061	13.6	997	9.8
node7.prof:	validateTransaction	1010	10.1	972	7.1
node7.prof:	txIsUnique	500	1.6	500	1.1
node7.prof:	nodeLogic.mint’	50	0.2	25	0.4
node8.prof:	nodeLogic.processTXs.processTXs’	1061	13.7	1035	11.3
node8.prof:	validateTransaction	1010	10.3	1010	8.1
node8.prof:	txIsUnique	505	1.4	498	1.3
node8.prof:	nodeLogic.mint’	50	1.3	24	0.3
node9.prof:	nodeLogic.processTXs.processTXs’	1061	14.1	986	11.1
node9.prof:	validateTransaction	1010	10.2	961	7.4
node9.prof:	txIsUnique	505	1.3	501	1.8
node9.prof:	nodeLogic.mint’	50	1.3	25	0.5

Για την μέτρηση του block time και της ρυθμαπόδοσης του συστήματος, λαμβάνονται υπόψιν τόσα μπλοκ όσα και οι κλήσεις στην συνάρτηση mint και τόσες συναλλαγές όσα τα μπλοκς επί την εκάστοτε χωρητικότητα.

$$\begin{aligned} &\text{Χωρητικότητα} = 5 \Rightarrow \text{Μπλοκς} = 100 \text{ και } \Sigma\text{υναλλαγές} = 500 \\ &\text{Χωρητικότητα} = 10 \Rightarrow \text{Μπλοκς} = 50 \text{ και } \Sigma\text{υναλλαγές} = 500 \\ &\text{Χωρητικότητα} = 20 \Rightarrow \text{Μπλοκς} = 25 \text{ και } \Sigma\text{υναλλαγές} = 500 \end{aligned} \tag{3}$$

3.3 Ρυθμαπόδοση και Block time

Στο σχήμα 6 φαίνονται οι μέσοι χρόνοι δημιουργίας block όπως υπολογίστηκαν από κάθε κόμβο.

```
capacity5/node10.log:Mean time between blocks: 550.0
capacity5/node1.log:Mean time between blocks: 550.0
capacity5/node2.log:Mean time between blocks: 550.0
capacity5/node3.log:Mean time between blocks: 550.0
capacity5/node4.log:Mean time between blocks: 550.0
capacity5/node5.log:Mean time between blocks: 550.0
capacity5/node6.log:Mean time between blocks: 550.0
capacity5/node7.log:Mean time between blocks: 322.2111111111111
capacity5/node8.log:Mean time between blocks: 322.2111111111111
capacity5/node9.log:Mean time between blocks: 550.0

capacity10/node10.log:Mean time between blocks: 1059.98
capacity10/node1.log:Mean time between blocks: 1059.98
capacity10/node2.log:Mean time between blocks: 1059.98
capacity10/node3.log:Mean time between blocks: 1059.98
capacity10/node4.log:Mean time between blocks: 1059.98
capacity10/node5.log:Mean time between blocks: 1059.98
capacity10/node6.log:Mean time between blocks: 1059.98
capacity10/node7.log:Mean time between blocks: 1059.98
capacity10/node8.log:Mean time between blocks: 1059.98
capacity10/node9.log:Mean time between blocks: 1059.98

capacity20/node10.log:Mean time between blocks: 1708.375
capacity20/node1.log:Mean time between blocks: 1708.375
capacity20/node2.log:Mean time between blocks: 1708.375
capacity20/node3.log:Mean time between blocks: 1708.375
capacity20/node4.log:Mean time between blocks: 1136.4545454545455
capacity20/node5.log:Mean time between blocks: 1708.375
capacity20/node6.log:Mean time between blocks: 1708.375
capacity20/node7.log:Mean time between blocks: 1708.375
capacity20/node8.log:Mean time between blocks: 1708.375
capacity20/node9.log:Mean time between blocks: 1708.375
```

Σχήμα 6: Μέσος χρόνος δημιουργίας block (ms)

Πάλι, από τους χρόνους αυτούς μπορούν να μετρηθούν και οι εξυπηρετούμενες συναλλαγές ανά δευτερόλεπτο. Για κάθε capacity έχουμε:

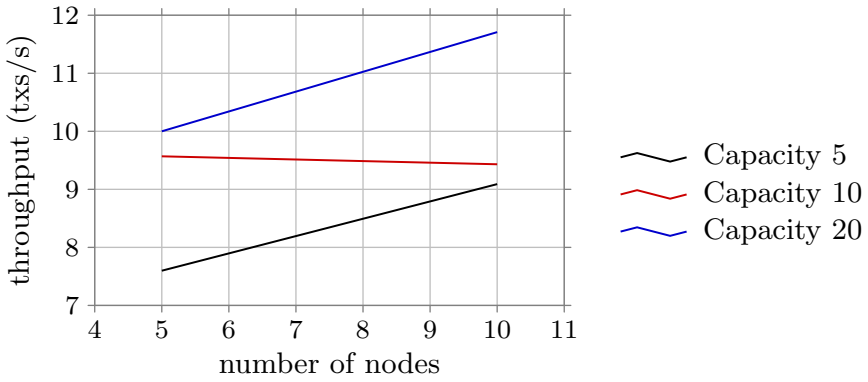
$$\text{Ρυθμαπόδοση} = \frac{\text{Συναλλαγές}}{\text{Χρόνος}} = \frac{\text{Συναλλαγές}}{\text{Μπλοκ}} \cdot \frac{\text{Μπλοκ}}{\text{Χρόνος}} \Leftrightarrow$$

$$\text{Ρυθμαπόδοση} = \frac{\frac{\text{Συναλλαγές}}{\text{Μπλοκ}}}{\text{Μέσος χρόνος δημιουργίας μπλοκ}} = \frac{\text{Χωρητικότητα}}{\text{Μέσος χρόνος δημιουργίας μπλοκ}} \Rightarrow \tag{4}$$

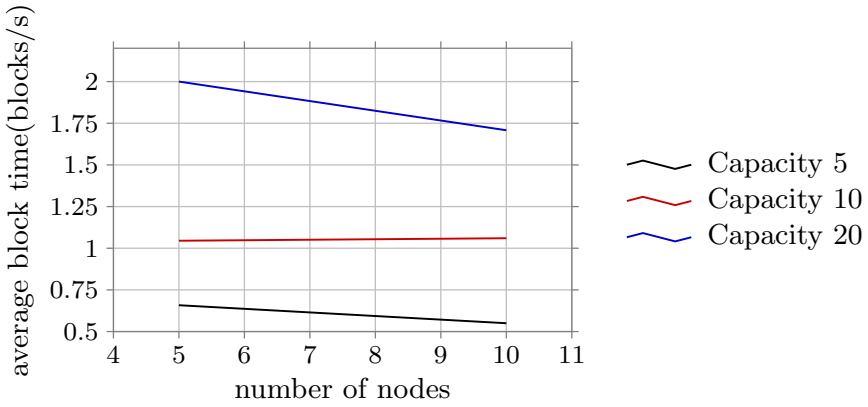
$$\boxed{\begin{aligned} \text{Ρυθμαπόδοση}_{capacity=5} &= \frac{5}{0,550} = 9,090\frac{txs}{s} \\ \text{Ρυθμαπόδοση}_{capacity=10} &= \frac{10}{1,060} = 9,4323\frac{txs}{s} \\ \text{Ρυθμαπόδοση}_{capacity=20} &= \frac{20}{1.708} = 11,709\frac{txs}{s} \end{aligned}}$$

Οι μετρήσεις από τα πειράματα συνοψίζονται στα γραφήματα 7.

(α΄) Throughput του συστήματος



(β΄) Μέσο block time του συστήματος

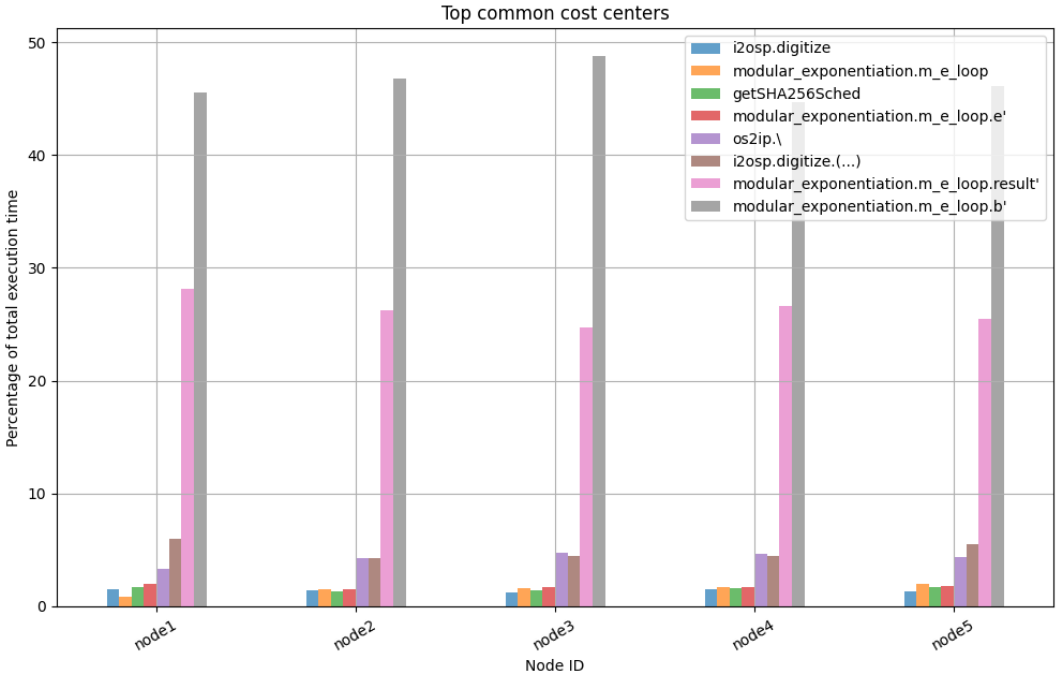


Σχήμα 7: Ρυθμαπόδοση και μέσος block time του συστήματος

Στα γραφήματα 7α΄ και 7β΄ φαίνεται η ρυθμαπόδοση και το μέσο block time του συστήματος μεταξύ των πειραμάτων, από 5 έως 10 κόμβους, για κάθε χωρητικότητα. Φαίνεται, ότι το πλήθος των εξυπηρετούμενων συναλλαγών ανά μονάδα χρόνου αυξάνεται με τον αριθμό των κόμβων για τις χωρητικότητες 5 και 20, αλλά για την 10 μένει οριακά σταθερός, με ελαφρώς φθίνουσα τάση. Ο δε μέσος χρόνος δημιουργίας block μειώνεται με τον αριθμό των κόμβων για τις χωρητικότητες 5 και 20, αλλά για την 10 μένει οριακά σταθερός, με ελαφρώς αύξουσα τάση. Αυτό δείχνει ότι το δίκτυο μπορεί να κλιμακώνει με τον αριθμό των κόμβων.

4 Δικαιοσύνη

Στο πείραμα δικαιοσύνης, το δίκτυο εκκινείται με 5 κόμβους και ο υπάριθμόν 1 από αυτούς κάνει stake 100 BCC, ενώ οι υπόλοιποι κάνουν stake 10 BCC.

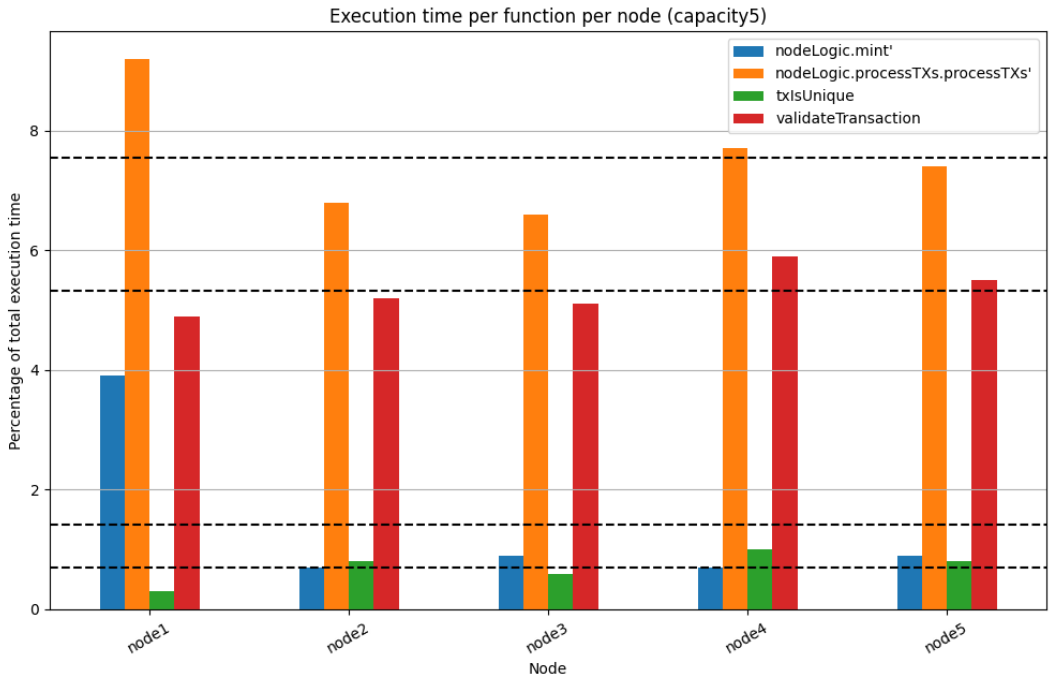


Σχήμα 8: Τα πιο χρονοβόρα κομμάτια του κώδικα capacity=5

Στον πίνακα 3 φαίνονται οι κλήσεις μερικών συναρτήσεων ενδιαφέροντος. Φαίνεται ότι τα πλήθη όλων των κλήσεων είναι ίδια ανά κόμβο, πράγμα που σημαίνει ότι οι κόμβοι εκτελούν τις ίδιες λειτουργίες με την ίδια συχνότητα. Παρ'όλα αυτά, ο κόμβος με το μεγαλύτερο stake καταναλώνει πολύ περισσότερο χρόνο στην συνάρτηση mint' σε σχέση με τους υπόλοιπους κόμβους, όπως φαίνεται και στο σχήμα 9.

Πίνακας 3: Στατιστικά συναρτήσεων ανά κόμβο capacity=5

Node	Function	Entries	TimeInh
node1.prof:	nodeLogic.processTXs.processTXs’	301	9.2
node1.prof:	validateTransaction	255	4.9
node1.prof:	txIsUnique	230	0.3
node1.prof:	nodeLogic.mint’	45	3.9
node2.prof:	nodeLogic.processTXs.processTXs’	301	6.8
node2.prof:	validateTransaction	255	5.2
node2.prof:	txIsUnique	226	0.8
node2.prof:	nodeLogic.mint’	45	0.7
node3.prof:	nodeLogic.processTXs.processTXs’	300	6.6
node3.prof:	validateTransaction	255	5.1
node3.prof:	txIsUnique	222	0.6
node3.prof:	nodeLogic.mint’	44	0.9
node4.prof:	nodeLogic.processTXs.processTXs’	300	7.7
node4.prof:	validateTransaction	255	5.9
node4.prof:	txIsUnique	223	1.0
node4.prof:	nodeLogic.mint’	44	0.7
node5.prof:	nodeLogic.processTXs.processTXs’	300	7.4
node5.prof:	validateTransaction	255	5.5
node5.prof:	txIsUnique	224	0.8
node5.prof:	nodeLogic.mint’	44	0.9



Σχήμα 9: Ποσοστό χρόνου επί του συνολικού χρόνου εκτέλεσης που λαμβάνει η κάθε συνάρτηση capacity=5

Στο σχήμα 9 φαίνεται, πράγματι, ότι ο κόμβος με το μεγαλύτερο stake καταναλώνει πολύ περισσότερο χρόνο στην συνάρτηση mint' σε σχέση με τους υπόλοιπους κόμβους, ενδεικτικό του γεγονός ότι πράγματι αυτός αναλαμβάνει συχνότερα την δημιουργία των νέων blocks. Μάλιστα, επισκοπώντας τα υπόλοιπα των λογαριασμών των κόμβων στο σχήμα 10, παρατηρεί κανείς ότι όντως τα περισσότερα νομίσματα συσσωρεύονται στον κόμβο με το μεγαλύτερο stake. Συμπεραίνεται, λοιπόν, ότι σε βάθος χρόνου συσσωρεύονται νομίσματα στον κόμβο με το μεγαλύτερο stake. Θεωρητικά, αυτό σημαίνει ότι ένας κακόβουλος κόμβος θα μπορούσε να εκμεταλλευτεί το φαινόμενο αυτό και να χειραγωγεί το δίκτυο κατά την θέλησή του. Επομένως, υπάρχει ανάγκη για έναν μηχανισμό που θα εξασφαλίζει ότι, παρά την ανισότητα των stakes, οι κόμβοι θα έχουν ίσες ευκαιρίες στην επικύρωση των συναλλαγών και δεν θα επαφίεται η ασφάλεια του δικτύου σε έναν μόνο κόμβο.

```
==> node1.log <==
> 3497.2

==> node2.log <==
> 280.70000000000005

==> node3.log <==
> 232.70000000000005

==> node4.log <==
> 191.70000000000005

==> node5.log <==
> 697.7
```

Σχήμα 10: Υπόλοιπα λογαριασμών κόμβων capacity=5