Υπολογιστικά Νέφη
ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3
« Σύστημα Δύο Παράλληλων Εξυπηρετητών και Εκ-Περιτροπής Ανάθεση »
Οκτώβριος 2024

Συμπληρώστε τα στοιχεία σας πριν την αποστολή της εργασίας	
Ονοματεώπυνο	Δημήτρης Γκούμας
Αριθμός Μητρώου	4502

Περιγραφή και Βασικά Χαρακτηριστικά των Μοντέλων Ουρών

Η έννοια των ουρών είναι πολύτιμη στην ανάλυση της απόδοσης των υπολογιστικών συστημάτων. **Οι ουρές παρέχουν τη δυνατότητα να εκτιμηθούν σπουδαίες ιδιότητες και χαρακτηριστικά μετρικά της απόδοσης των υπολογιστικών/επεξεργαστικών συστημάτων**, όπως είναι για παράδειγμα ο χρόνος απόκρισης ή η ρυθμο-απόδοση και το πλήθος των εξυπηρετούμενων tasks.

Οι ουρές που αποθηκεύουν τα γεγονότα (όπως για παράδειγμα τις εντολές που φτάνουν στον επεξεργαστή ή τις εφαρμογές) συνεπάγονται τυχαία (ή στοχαστική) συμπεριφορά, αφού προσπαθούν να προσεγγίσουν τη συμπεριφορά και το ρυθμό με τον οποίο παράγονται τα γεγονότα, κάτι που συνήθως κάνει τη μαθηματική ανάλυση και τις τεχνικές πρόβλεψης δύσκολες και απαιτητικές. Για το λόγο αυτό συχνά προτιμάται η μελέτη της απόδοσης μέσω προσομοιώσεων διακριτών γεγονότων, ώστε αν γίνει κατανοητή η επίδοση του συστήματος.

Για τη μελέτη της ουράς, θεωρούμε πρώτα ένα σύνολο εργασιών περιμένουν σε μια ουρά εξυπηρέτηση από το κεντρικό επεξεργαστικό σύστημα (CPU). Η CPU μπορεί να αποτελείται από πολλαπλούς επεξεργαστές κάθε ένας από τους οποίους μπορεί να εξυπηρετήσει μια εργασία.

Αν όλοι οι επεξεργαστές είναι απασχολημένοι, τότε οι εργασίες που φθάνουν περιμένουν σε μία ουρά. Στην ορολογία των ουρών, οι εργασίες ονομάζονται επίσης και “πελάτες”.

Για να αναλυθεί ένα τέτοιο σύστημα, θα πρέπει να καθορισθούν τα επόμενα χαρακτηριστικά του συστήματος:

- **Διαδικασία άφιξης:** Εάν οι εργασίες φθάνουν σε χρόνους t1, t2,…,tj, τότε οι τυχαίες μεταβλητές tj,= tj - tj-1 ονομάζονται μεταξύ των αφίξεων χρόνοι (interarrival times). Οι τυχαίες μεταβλητές αποτελούν μια σειρά ανεξάρτητων και ομοιόμορφα κατανεμημένων (Independent and Identically Distributed, IID) τυχαίων μεταβλητών. Η πιο γνωστή διαδικασία άφιξης είναι η γνωστή ως αφίξεις Poisson (Poisson arrivals), η οποία σημαίνει ότι οι μεταξύ των αφίξεων χρόνοι είναι IID και είναι εκθετικά κατανεμημένοι. Χρησιμοποιούνται βέβαια και άλλες κατανομές όπως είναι η Erlang

και η υπερεκθετική. Στην πραγματικότητα, πολλά αποτελέσματα ουρών ισχύουν για όλες τις κατανομές των χρόνων άφιξης. Σε μια τέτοια περίπτωση, λέμε ότι τα αποτελέσματα ισχύουν για μια γενική (general) κατανομή.

- **Κατανομή του χρόνου εξυπηρέτησης:** Χρειάζεται επίσης να γνωρίζουμε το χρόνο που η κάθε εργασία εξυπηρετείται από τη CPU. Ο χρόνος αυτός ονομάζεται χρόνος εξυπηρέτησης. Οι χρόνοι εξυπηρέτησης είναι τυχαίες μεταβλητές και είναι IID. Η συχνότερα χρησιμοποιούμενη κατανομή είναι η εκθετική. Χρησιμοποιούνται όμως και άλλες κατανομές όπως είναι η Erlang, η υπερεκθετική και η γενική. Τα αποτελέσματα μιας γενικής κατανομής ισχύουν για όλες τις κατανομές του χρόνου εξυπηρέτησης.
- **Αριθμός των εξυπηρετών:** Η CPU μπορεί να έχει έναν ή περισσότερους επεξεργαστές οι οποίοι θεωρούνται σαν τμήμα του ίδιου συστήματος της ουράς επειδή είναι όλοι ίδιοι. Κάθε επεξεργαστής μπορεί να εξυπηρετήσει οποιαδήποτε εργασία. Εάν οι εξυπηρέτες δεν είναι όλοι ίδιοι, συνήθως διαιρούνται σε ομάδες ίδιων εξυπηρετών με ξεχωριστές ουρές σε κάθε ομάδα. Τότε κάθε ομάδα είναι ένα σύστημα ουράς.
- **Χωρητικότητα του συστήματος:** Ο μέγιστος αριθμός των προγραμμάτων που μπορούν να περιμένουν στην ουρά μπορεί να είναι περιορισμένος. Ο αριθμός αυτός ονομάζεται χωρητικότητα του συστήματος. Στα περισσότερα συστήματα η χωρητικότητα είναι πεπερασμένη. Αν ο αριθμός είναι μεγάλος, τότε είναι πιο εύκολο να γίνει η ανάλυση αν υποτεθεί απεριόριστη χωρητικότητα. Η χωρητικότητα του συστήματος περιλαμβάνει και τις εργασίες που περιμένουν εξυπηρέτηση και εκείνες που δέχονται εξυπηρέτηση.
- **Μέγεθος πληθυσμού:** Ο συνολικός αριθμός των πιθανών εργασιών που πρόκειται να έρθουν στο σύστημα είναι το μέγεθος πληθυσμού. Στα περισσότερα πραγματικά συστήματα το μέγεθος πληθυσμού είναι πεπερασμένο. Αν το μέγεθος αυτό είναι μεγάλο, τότε η ανάλυση είναι πιο εύκολη αν θεωρηθεί ότι το μέγεθος είναι απεριόριστο.
- **Πειθαρχία εξυπηρέτησης της ουράς:** Η σειρά με την οποία εξυπηρετούνται οι εργασίες είναι ονομάζεται πειθαρχία εξυπηρέτησης.
    - FCFS (First Come First Served) ή αλλιώς και FIFO (First In - First Out).
    - LCFS (Last Come First Served) ή αλλιώς και LIFO (Last In - First Out).
    - LCFS-PR (Last Come First Served with Preempt and Resume) η οποία μπορεί αν σταματήσει μία εκτέλεση των tasks για να δώσει priority σε ένα task της ουράς που έχει μεγαλύτερο priority
- **Τεχνικές εξισορρόπησης φόρτου (Load Balancing) – Ανάθεση σε επεξεργαστή (Λειτουργία dispatcher):** Συνήθως οι επεξεργαστές χρησιμοποιούν την ανάθεση εργασιών εκ περιτροπής RR (Round Robin) με σταθερό κβάντο, δηλαδή την σειριακή ανάθεση ενός σταθερού αριθμού από tasks σε κάθε επόμενο επεξεργαστή.

Εάν το μέγεθος του κβάντου είναι μικρό σε σχέση με το μέσο χρόνο εξυπηρέτησης, τότε η RR τεχνική ανάθεσης ονομάζεται ***πειθαρχία διαμοιρασμού των επεξεργαστών (PS,***

Processor Sharing), επειδή κάθε μια από τις n εργασίες που περιμένουν παίρνει το 1/n του χρόνου του επεξεργαστή.

Ένα σύστημα με σταθερή καθυστέρηση, ονομάζεται απεριόριστος εξυπηρέτης (Infinite Server, IS) ή κέντρο καθυστέρησης (delay center). Συνήθως τα τερματικά στα συστήματα διαμοιρασμού χρόνου μοντελοποιούνται σαν κέντρα καθυστέρησης.

Μερικές φορές **ο σχεδιασμός ανάθεσης σε επεξεργαστή και τεχνικών εξισορρόπησης βασίζεται στο χρόνο εξυπηρέτησης που απαιτείται**. Παραδείγματα τέτοιων πειθαρχιών είναι να σχεδιάζεται πρώτη η εργασία:

- με το μικρότερο χρόνο επεξεργασίας *(Shortest Processing Time First, SPT)*,
- με το μικρότερο υπολειπόμενο χρόνο επεξεργασίας *(Shortest Remaining Processing Time First, SRPT)*,
- με το μικρότερο αναμενόμενο χρόνο επεξεργασίας *(Shortest Expected Processing Time first, SEPT)*, και
- με το μικρότερο αναμενόμενο υπολειπόμενο χρόνο επεξεργασίας *(Shortest Expected Remaining Processing Time first, SEPT)*.

Ορισμός ενός συστήματος ουρών:

Συμβολισμός του Kendall: A/S/m/B/K/SD

A	είναι η κατανομή των μεταξύ των αφίξεων χρόνων,
S	είναι η κατανομή των χρόνων εξυπηρέτησης,
m	είναι ο αριθμός των εξυπηρετών,
B	δηλώνει την χωρητικότητα του συστήματος,
K	είναι το μέγεθος πληθυσμού, και
SD	είναι η πειθαρχία εξυπηρέτησης.

Οι κατανομές για τους μεταξύ των αφίξεων χρόνους και τους χρόνους εξυπηρέτησης δηλώνονται με ένα γράμμα:

M	Εκθετική
E_k	Erlang-k
H_k	Υπερεκθετική με παράμετρο k
D	Ντετερμινιστική
G	Γενική

Μια ντετερμινιστική κατανομή συνεπάγεται ότι οι χρόνοι είναι σταθεροί και ότι δεν υπάρχει διασπορά. **Η γενική κατανομή** σημαίνει ότι δεν ορίζεται η κατανομή και ότι τα αποτελέσματα ισχύουν για όλες τις κατανομές.

Θα πρέπει να σημειωθεί ότι η εκθετική κατανομή δηλώνεται με M λόγω της λέξης memoryless.

Αν οι μεταξύ των αφίξεων χρόνοι είναι εκθετικά κατανεμημένοι, με μέση τιμή για παράδειγμα 1/λ, τότε **ο αναμενόμενος χρόνος για την επόμενη άφιξη είναι πάντα 1/λ άσχετα με το χρόνο που πέρασε** από την προηγούμενη άφιξη. Αυτή είναι η "χωρίς μνήμη" ιδιότητα μόνο της εκθετικής κατανομής, και γιαυτό **ονομάζεται κατανομή χωρίς μνήμη (memoryless distribution).**

Οι μαζικές αφίξεις (bulk arrivals) και η μαζική εξυπηρέτηση (bulk service), όπου κάθε άφιξη ή εξυπηρέτηση αποτελείται από μια ομάδα εργασιών, δηλώνονται με έναν εκθέτη.
Μαζικές Poisson αφίξεις ή εξυπηρετήσεις δηλώνονται με $M^{[x]}$, όπου το x παριστάνει το μέγεθος της ομάδας, που είναι γενικά μια τυχαία μεταβλητή της οποίας η κατανομή πρέπει να ορισθεί ξεχωριστά. Η συγκεκριμένη κατανομή των μαζικών αφίξεων συχνά ονομάζεται αναφέρεται και ως αφίξεις κατά ρίπους (κατά ριπάς, burst mode arrivals). Με $G^{[x]}$ παριστάνεται μια μαζική διαδικασία άφιξης ή εξυπηρέτησης με γενικούς μεταξύ των ομάδων χρόνους (άφιξης ή εξυπηρέτησης αντίστοιχα).

Παράδειγμα 1:
Ο συμβολισμός M/M/2/16/1000/LCFS δηλώνει ένα σύστημα μιας ουράς με τις ακόλουθες παραμέτρους:

1. Οι μεταξύ των αφίξεων χρόνοι είναι εκθετικά κατανεμημένοι.
2. Οι χρόνοι εξυπηρέτησης είναι εκθετικά κατανεμημένοι.
3. Υπάρχουν δύο εξυπηρέτες.
4. Η ουρά έχει χώρο για 16 εργασίες. Από τις 16 θέσεις οι δύο είναι για τις εργασίες που εξυπηρετούνται και οι 14 για τις εργασίες που περιμένουν εξυπηρέτηση. Όταν ο αριθμός των εργασιών γίνει 16, τότε όλες οι εργασίες χάνονται έως ότου ελαττωθεί το μήκος της ουράς.
5. Υπάρχει ένα σύνολο 1000 εργασιών που μπορούν να εξυπηρετηθούν.
6. Η πειθαρχία εξυπηρέτησης είναι αυτός που έρχεται τελευταίος εξυπηρετείται πρώτος.

Αν δεν δηλωθεί ρητώς, οι ουρές θεωρείται ότι έχουν απεριόριστη χωρητικότητα, απεριόριστο μέγεθος πληθυσμού, και πειθαρχία εξυπηρέτησης FCFS.

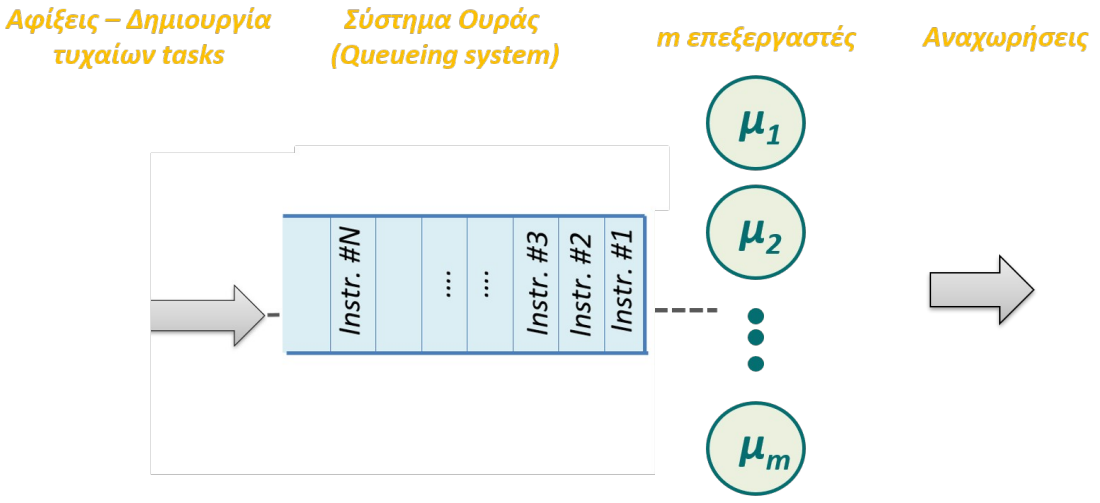
Συνήθως, μόνον οι πρώτες τρεις από τις έξι παραμέτρους είναι αρκετές για να δηλώσουν τον τύπο της ουράς. Για παράδειγμα, η ουρά $M/M/2/\infty/\infty$/FCFS δηλώνεται σαν M/M/2.

Παράδειγμα 2
Ο συμβολισμός $M^{[x]}/M/1$ δηλώνει μια ουρά με έναν εξυπηρέτη, με μαζικές αφίξεις Poisson και με εκθετικούς χρόνους εξυπηρέτησης. Ο συμβολισμός $M/G^{[x]}/m$ δηλώνει μια ουρά με διαδικασία άφιξης
Poisson, μαζική εξυπηρέτηση με γενική κατανομή χρόνων εξυπηρέτησης, και με m εξυπηρέτες.

Στην παρούσα παράγραφο, περιγράφονται μερικά (εισαγωγικά) χαρακτηριστικά και κανόνες που ισχύουν για όλες τις ουρές στους επεξεργαστές. Οι χαρακτηρικές τιμές και βασικά μετρικά απόδοσης είναι τα ακόλουθα:

- τ = Μεταξύ των αφίξεων χρόνος.
- λ = μέσος ρυθμός άφιξης = 1 / $E[\tau]$. Σε μερικά συστήματα, ο χρόνος αυτός μπορεί να είναι
  συνάρτηση της κατάστασης του συστήματος. Για παράδειγμα, μπορεί να εξαρτάται από
  τον αριθμό των εργασιών που υπάρχουν ήδη στο σύστημα.
- s = χρόνος εξυπηρέτησης ανά εργασία.
- μ = μέσος ρυθμός εξυπηρέτησης ανά εξυπηρέτη, = 1 / $E[s]$. Ο συνολικός ρυθμός εξυπηρέτησης για τους m εξυπηρέτες είναι $m\mu$.
- n = αριθμός των εργασιών στο σύστημα, δηλαδή μήκος ουράς (queue length). Ο αριθμός
  αυτός περιλαμβάνει τις εργασίες που περιμένουν στην ουρά καθώς και τις εργασίες που
  δέχονται εξυπηρέτηση.
- n_q = αριθμός των εργασιών που περιμένουν εξυπηρέτηση. Είναι προφανές ότι ο αριθμός
  αυτός είναι μικρότερος του n.
- n_s = αριθμός των εργασιών που δέχονται εξυπηρέτηση.
- r = χρόνος απόκρισης ή χρόνος στο σύστημα (περιλαμβάνει το χρόνο που περιμένει μια
  εργασία για εξυπηρέτηση και το χρόνο που δέχεται εξυπηρέτηση).
- w = χρόνος αναμονής, δηλαδή το διάστημα μεταξύ του χρόνου άφιξης και της στιγμής που
  ξεκινάει η εξυπηρέτηση.

Το παρακάτω σχήμα απεικονίζει ένα σύστημα με άπειρες θέσεις μνήμης (δεν κλείνει/τερματίζεται το τέλος της ουράς με κάθετη γραμμή) και m παράλληλους επεξεργαστές με ρυθμός επεξεργασίας μ.



Εικόνα 1 Σύστημα με άπειρες θέσεις μνήμης (δεν κλείνει/τερματίζεται το τέλος της ουράς με κάθετη γραμμή) και m παράλληλους επεξεργαστές με διαφορετικούς ρυθμός επεξεργασίας σε μία κεντρική ουρά (central queue)

Εκτός από τα λ και μ όλες οι άλλες μεταβλητές είναι τυχαίες μεταβλητές. Υπάρχουν ορισμένες σχέσεις μεταξύ των μεταβλητών αυτών που ισχύουν στις G/G/m ουρές.

Συνθήκη σταθερότητας.
Εάν ο αριθμός των προγραμμάτων στο σύστημα αυξάνει συνεχώς και γίνεται άπειρος, τότε το σύστημα λέγεται ασταθές. Για σταθερότητα, ο μέσος ρυθμός άφιξης θα πρέπει να είναι μικρότερος από το μέσο ρυθμό εξυπηρέτησης:

$$\lambda < m\,\mu$$

Αυτή η σχέση της σταθερότητας δεν ισχύει στην περίπτωση πεπερασμένου πληθυσμού και στην περίπτωση συστημάτων με πεπερασμένο πλήθος θέσεων στην ουρά.

Στα συστήματα πεπερασμένου πληθυσμού, το μήκος της ουράς είναι πάντα πεπερασμένο και συνεπώς το σύστημα δεν μπορεί ποτέ να γίνει ασταθές. Επίσης τα συστήματα με περιορισμένο αριθμό θέσεων στην ουρά είναι πάντα ευσταθή επειδή οι αφίξεις χάνονται όταν ο αριθμός των εργασιών στο σύστημα υπερβαίνει των αριθμό των θέσεων, δηλαδή τη χωρητικότητα της ουράς.

Αριθμός στο σύστημα σε σχέση με τον αριθμό στην ουρά.
Ο αριθμός των εργασιών στο σύστημα είναι πάντα ίσος με το άθροισμα του αριθμού των εργασιών στην ουρά και του αριθμού των εργασιών που δέχονται εξυπηρέτηση:

$$n = n_q + n_s$$

Τα *n*, *nq*, και *ns* είναι τυχαίες μεταβλητές. Η ισότητα αυτή οδηγεί στην επόμενη σχέση μεταξύ των μέσων τιμών τους.

$$E[n] = E[n_q] + E[n_s]$$

Δηλαδή, ο μέσος αριθμός των εργασιών στο σύστημα είναι ίσος με το άθροισμα του μέσου αριθμού στην ουρά και του μέσου αριθμού σε εξυπηρέτηση.

Αριθμός σε σχέση με το χρόνο.
Αν δεν χάνονται εργασίες λόγω μή επαρκών θέσεων στην ουρά, τότε:

Μέσος αριθμός εργασιών στο σύστημα = ρυθμός άφιξης × μέσος χρόνος απόκρισης (Εξ. 1)

Όμοια:
Μέσος αριθμός εργασιών στην ουρά = ρυθμός άφιξης × μέσος χρόνος αναμονής (Εξ. 2)

Οι εξισώσεις (1) και (2) είναι γνωστές σαν κανόνας του Little (Little's law).

Σε συστήματα με πεπερασμένη χωρητικότητα ο κανόνας αυτός μπορεί να χρησιμοποιηθεί με την προϋπόθεση ότι θα χρησιμοποιηθεί ο αποτελεσματικός ρυθμός άφιξης, δηλαδή ο

ρυθμός των προγραμμάτων που πραγματικά μπαίνουν στο σύστημα και δέχονται εξυπηρέτηση.

Η σχέση αυτή ισχύει σε όλα τα συστήματα ή μέρη των συστημάτων όπου ο αριθμός των εργασιών που μπαίνουν στο σύστημα είναι ίσος με τον αριθμό των εργασιών που τελειώνουν την εξυπηρέτηση.

Ο κανόνας του Little που αποδείχθηκε από τον Little το 1961 βασίζεται σε μια άποψη του συστήματος σαν να είναι ένα “μαύρο κουτί”.

Ο κανόνας ισχύει όσο ο αριθμός των εργασιών που μπαίνουν στο σύστημα είναι ίσος με τον αριθμό των εργασιών που τελειώνουν εξυπηρέτηση, χωρίς να δημιουργούνται νέες εργασίες στο σύστημα και χωρίς ποτέ να χάνονται εργασίες στο σύστημα.

Ακόμη και σε συστήματα στα οποία μερικές εργασίες χάνονται λόγω της πεπερασμένης χωρητικότητας στις ουρές, ο κανόνας μπορεί να εφαρμοσθεί στο μέρος του συστήματος που αποτελείται από τις θέσεις αναμονής και εξυπηρέτησης γιατί από τη στιγμή που μια εργασία βρίσκει μια θέση αναμονής, δεν χάνεται.

Ο ρυθμός άφιξης στην περίπτωση αυτή θα πρέπει να ρυθμιστεί με τρόπο ώστε να μην περιλαμβάνονται οι εργασίες που χάνονται πριν βρουν μια θέση στην ουρά. Δηλαδή, θα πρέπει να χρησιμοποιείται ο αποτελεσματικός ρυθμός άφιξης των εργασιών που μπαίνουν στο σύστημα.

Χρόνος στο σύστημα και χρόνος στην ουρά.
Ο χρόνος κατά τον οποίο μια εργασία βρίσκεται σε ένα σύστημα ουρών είναι ίσος με το άθροισμα του χρόνου που περιμένει στην ουρά και το χρόνο που δέχεται εξυπηρέτηση, δηλαδή ισχύει:

$$r = w + s$$

όπου r, w, και s είναι τυχαίες μεταβλητές. Η σχέση αυτή οδηγεί στην επόμενη:

$$E[r] = E[w]+E[s]$$

Δηλαδή, ο μέσος χρόνος απόκρισης είναι ίσος με το άθροισμα του μέσου χρόνου αναμονής και του μέσου χρόνου εξυπηρέτησης.

Παράδειγμα 3.
Ενας παρακολουθητής σε έναν εξυπηρέτη δίσκου έδειξε ότι ο μέσος χρόνος για την εξυπηρέτηση μιας I/O αίτησης είναι 100 ms. Ο ρυθμός I/O ήταν περίπου 100 αιτήσεις ανά δευτερόλεπτο. Ποιός ήταν ο μέσος αριθμός αιτήσεων στον εξυπηρέτη του δίσκου;

Σύμφωνα με τον κανόνα του Little: Μέσος αριθμός στον εξυπηρέτη του δίσκου = ρυθμός άφιξης ×
χρόνος απόκρισης = (100 αιτήσεις ανά δευτερόλεπτο) × (0.1 δευτερόλεπτα) = 10 αιτήσεις

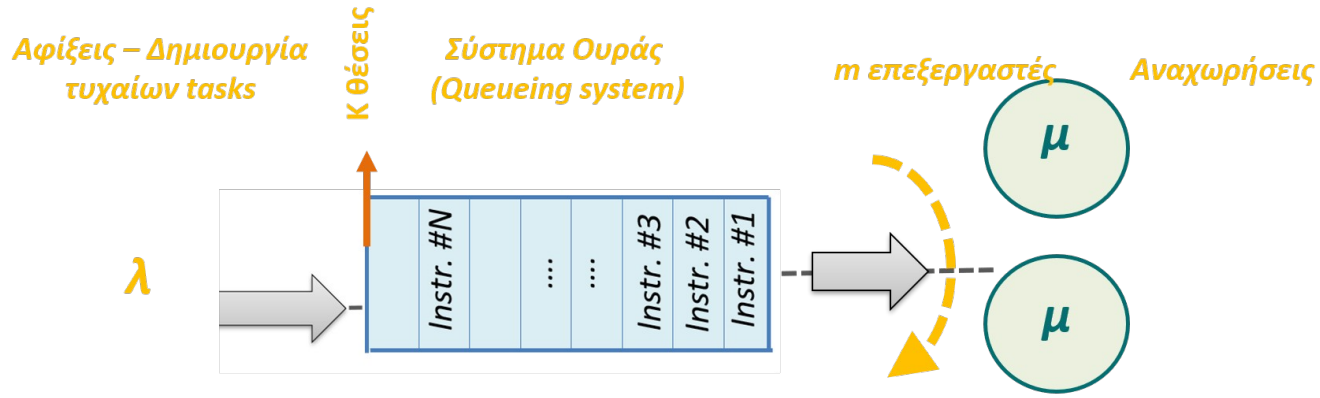
Η παρούσα σειρά εργαστηριακών ασκήσεων έχει στόχο αν μελετήσει και αν συγκρίνει τρεις περιπτώσεις συστημάτων με δύο παράλληλους επεξεργαστές:

- **Lab3a:** Συμμετρικό σύστημα δύο παράλληλων επεξεργαστών με μία κοινή ουρά
- **Lab3b:** Συμμετρικό σύστημα δύο παράλληλων επεξεργαστών ο καθένας με τη δική του ουρά
- **Lab3c:** Ασύμμετρο σύστημα δύο παράλληλων επεξεργαστών και Round Robin ανάθεση

1η σειρά από ερωτήματα για συμμετρικό σύστημα δύο παράλληλων επεξεργαστών με μία κοινή ουρά :

Στην παρούσα άσκηση και σειρά από ερωτήματα θα μελετηθεί ένα συμμετρικό σύστημα δύο παράλληλων επεξεργαστών με μία κοινή ουρά, όπως απεικονίζεται στο παρακάτω σύστημα. Να προσομοιώσετε το παρακάτω σύστημα για τις εξής τιμές:

- Άφιξη εργασιών με εκθετική κατανομή και ρυθμό άφιξης λ=5
- Δύο παράλληλοι επεξεργαστές με εκθετική κατανομή και ρυθμό επεξεργασίας μ = 6
- Κοινή ουρά με K= 10 θέσεις
- Επιλογή επεξεργαστή εκ-περιτροπής, δηλαδή μία εργασία στον έναν και μια στον άλλον κυκλικά
- Χρόνος προσομοίωσης t = 1000



Εικόνα 2 Σύστημα δύο επεξεργαστών με μία κοινή ουρά

1) Να συμπεριλάβετε εδώ τον κώδικά σας για την παραπάνω προσομοίωση και να παρουσιάσετε τα αποτελέσματα για
   a. Το μέσο χρόνο αναμονής στην ουρά των tasks.
   b. Τον συνολικό-αθροιστικό αριθμό αφίξεων νέων tasks στο σύστημα.
   c. Το συνολικό-αθροιστικό αριθμό αναχωρήσεων-εξυπηρετήσεων των tasks στο σύστημα.
   d. Το πλήθος των εργασιών που έχουν εξυπηρετηθεί από το πρώτο επεξεργαστή, το δεύτερο επεξεργαστή και συνολικά από το σύστημα
   d. Το μέσο συντελεστή χρησιμοποίησης του κάθε επεξεργαστή - ποσοστό utilization.
   e. Το πλήθος των tasks που έχουν γίνει dropped

ΑΠΑΝΤΗΣΗ:

```
import random

arrivalRate = 5
serviceRate = 6
queueSize = 10
simulationTime = 1000

queue = []
p1BusyUntilTime = float('inf')
p2BusyUntilTime = float('inf')
totalArrivals = 0
totalDepartures = 0
droppedRequests = 0
totalServedByP1 = 0
totalServedByP2 = 0
totalWaitTime = 0

currentTime = 0
nextArrival = random.expovariate(arrivalRate)

while currentTime < simulationTime:
currentTime = nextArrival

if p1BusyUntilTime ⩽ currentTime:
p1BusyUntilTime = float('inf')
totalServedByP1 += 1
totalDepartures += 1
if p2BusyUntilTime ⩽ currentTime:
p2BusyUntilTime = float('inf')
totalServedByP2 += 1
totalDepartures += 1

if currentTime == nextArrival:
totalArrivals += 1

if len(queue) < queueSize:
queue.append(currentTime)
else:
droppedRequests += 1

nextArrival = currentTime + random.expovariate(arrivalRate)

if queue:
if p1BusyUntilTime == float('inf'):
p1BusyUntilTime = currentTime + random.expovariate(serviceRate)
totalWaitTime += currentTime - queue.pop(0)

elif p2BusyUntilTime == float('inf'):
p2BusyUntilTime = currentTime + random.expovariate(serviceRate)
totalWaitTime += currentTime - queue.pop(0)

print(f"Average waiting time in the queue: {round(totalWaitTime / totalDepartures, 2)}")
print(f"Total number of arrivals: {totalArrivals}")
print(f"Total number of departures: {totalDepartures}")
print("Total number of requests served by each processor:")
print(f" Processor 1: {totalServedByP1}")
print(f" Processor 2: {totalServedByP2}")
print("Average utilization of each processor:")
print(f" Processor 1: {round(totalServedByP1 / (simulationTime * serviceRate) * 100, 2)}%")
print(f" Processor 2: {round(totalServedByP2 / (simulationTime * serviceRate) * 100, 2)}%")
```

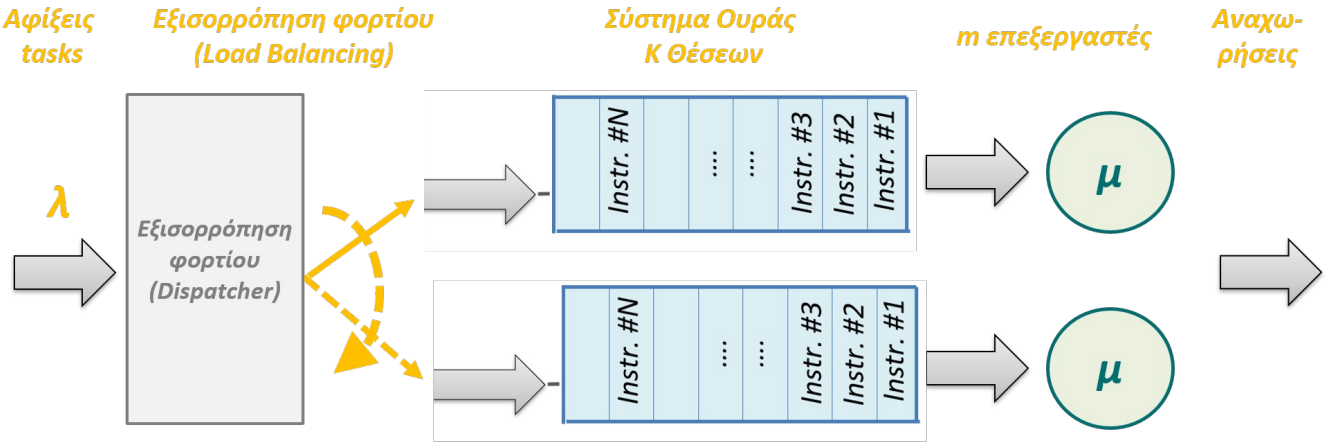
```
print(f"Number of dropped requests: {droppedRequests} ({round(droppedRequests / totalArrivals * 100, 2)}%)")
```

```
Average waiting time in the queue: 2.16
Total number of arrivals: 5034
Total number of departures: 4198
Total number of requests served by each processor:
  Processor 1: 2702
  Processor 2: 1496
Average utilization of each processor:
  Processor 1: 45.03%
  Processor 2: 24.93%
Number of dropped requests: 825 (16.39%)
```

2η σειρά από ερωτήματα για συμμετρικό σύστημα δύο παράλληλων επεξεργαστών ο καθένας με τη δική του ουρά :

Στην παρούσα άσκηση και σειρά από ερωτήματα θα μελετηθεί ένα συμμετρικό σύστημα δύο παράλληλων επεξεργαστών ο καθένας με τη δική του ουρά, όπως απεικονίζεται στην παρακάτω εικόνα για το σύστημα. Να προσομοιώσετε το παρακάτω σύστημα για τις εξής τιμές:

- Άφιξη εργασιών με εκθετική κατανομή και ρυθμό άφιξης λ=5
- Δύο παράλληλοι επεξεργαστές που ο καθένας ακολουθεί εκθετική κατανομή και ρυθμό επεξεργασίας μ = 6
- Κάθε επεξεργαστής έχει τη δική του ουρά με K= 10 θέσεις μνήμης
- Εξισορρόπηση φορτίου μέσω εκ-περιτροπής επιλογή επεξεργαστή (round robin), δηλαδή μία εργασία στην ουρά του πρώτου επεξεργαστή και μια εργασία στην ουρά του άλλου κυκλικά
- Χρόνος προσομοίωσης t = 1000



Εικόνα 3 Συμμετρικό σύστημα δύο παράλληλων επεξεργαστών ο καθένας με τη δική του ουρά

2) Να συμπεριλάβετε εδώ τον κώδικά σας για την παραπάνω προσομοίωση και να παρουσιάσετε τα αποτελέσματα για
   a. Το μέσο χρόνο αναμονής στην ουρά των tasks.
   b. Τον συνολικό-αθροιστικό αριθμό αφίξεων νέων tasks στο σύστημα.
   c. Το συνολικό-αθροιστικό αριθμό αναχωρήσεων-εξυπηρετήσεων των tasks στο σύστημα.
   d. Το πλήθος των εργασιών που έχουν εξυπηρετηθεί από το πρώτο επεξεργαστή, το δεύτερο επεξεργαστή και συνολικά από το σύστημα
   d. Το μέσο συντελεστή χρησιμοποίησης του κάθε επεξεργαστή - ποσοστό utilization.
   e. Το πλήθος των tasks που έχουν γίνει dropped από την πρώτη ουρά, τη δεύτερη ουρά και συνολικά από το σύστημα

ΑΠΑΝΤΗΣΗ:

```
import random

arrivalRate = 5
serviceRate = 6
queueSize = 10
simulationTime = 1000

queue1 = []
queue2 = []
p1BusyUntilTime = float('inf')
p2BusyUntilTime = float('inf')
totalArrivals = 0
totalDepartures = 0
droppedByP1 = 0
droppedByP2 = 0
totalDroppedRequests = 0
totalServedByP1 = 0
totalServedByP2 = 0
totalWaitTime = 0

currentTime = 0
nextArrival = random.expovariate(arrivalRate)

while currentTime < simulationTime:
currentTime = nextArrival

if p1BusyUntilTime ⩽ currentTime:
p1BusyUntilTime = float('inf')
totalServedByP1 += 1
totalDepartures += 1
if p2BusyUntilTime ⩽ currentTime:
p2BusyUntilTime = float('inf')
totalServedByP2 += 1
totalDepartures += 1

if currentTime == nextArrival:
totalArrivals += 1

if p1BusyUntilTime < p2BusyUntilTime:
if len(queue1) < queueSize:
queue1.append(currentTime)
else:
droppedByP1 += 1
elif p2BusyUntilTime < p1BusyUntilTime:
if len(queue2) < queueSize:
queue2.append(currentTime)
else:
droppedByP2 += 1

else:
if len(queue1) < len(queue2):
if len(queue1) < queueSize:
queue1.append(currentTime)
else:
droppedByP1 += 1
else:
if len(queue2) < queueSize:
queue2.append(currentTime)
else:
droppedByP2 += 1
```

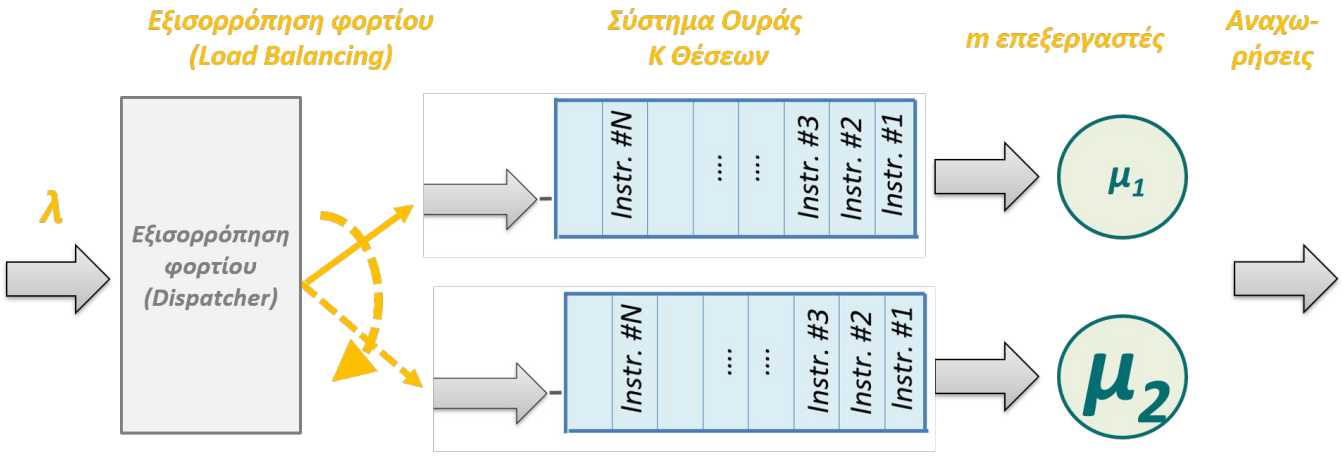
```
nextArrival = currentTime + random.expovariate(arrivalRate)

if queue1:
if p1BusyUntilTime == float('inf'):
p1BusyUntilTime = currentTime + random.expovariate(serviceRate)
totalWaitTime += currentTime - queue1.pop(0)

if queue2:
if p2BusyUntilTime == float('inf'):
p2BusyUntilTime = currentTime + random.expovariate(serviceRate)
totalWaitTime += currentTime - queue2.pop(0)
totalDroppedRequests = droppedByP1 + droppedByP2

print(f"Average waiting time in the queue: {round(totalWaitTime / totalDepartures, 2)}")
print(f"Total number of arrivals: {totalArrivals}")
print(f"Total number of departures: {totalDepartures}")
print("Total number of requests served by each processor:")
print(f" Processor 1: {totalServedByP1}")
print(f" Processor 2: {totalServedByP2}")
print("Average utilization of each processor:")
print(f" Processor 1: {round(totalServedByP1 / (simulationTime * serviceRate) * 100, 2)}%")
print(f" Processor 2: {round(totalServedByP2 / (simulationTime * serviceRate) * 100, 2)}%")
print("Number of dropped requests:")
print(f" Processor 1: {droppedByP1}")
print(f" Processor 2: {droppedByP2}")
print(f" Total: {totalDroppedRequests} ({round(totalDroppedRequests / totalArrivals * 100,
2)}%)")
```

```
Average waiting time in the queue: 0.85
Total number of arrivals: 5090
Total number of departures: 5073
Total number of requests served by each processor:
  Processor 1: 2411
  Processor 2: 2662
Average utilization of each processor:
  Processor 1: 40.18%
  Processor 2: 44.37%
Number of dropped requests:
  Processor 1: 6
  Processor 2: 3
  Total: 9 (0.18%)
```

3η σειρά από ερωτήματα για ασύμμετρο σύστημα δύο παράλληλων επεξεργαστών και Round Robin ανάθεση :

Στην παρούσα άσκηση και σειρά από ερωτήματα θα μελετηθεί ένα ασύμμετρο σύστημα δύο παράλληλων επεξεργαστών με διαφορετικούς ρυθμούς επεξεργασίας και ο καθένας με τη δική του ουρά, όπως απεικονίζεται στην παρακάτω εικόνα για το σύστημα. Να προσομοιώσετε το παρακάτω σύστημα για τις εξής τιμές:

- Άφιξη εργασιών με εκθετική κατανομή και ρυθμό άφιξης λ=5
- Δύο παράλληλοι επεξεργαστές που ο καθένας ακολουθεί εκθετική κατανομή και ρυθμούς επεξεργασίας $\mu_1 = 2$ και $\mu_2 = 8$
- Κάθε επεξεργαστής έχει τη δική του ουρά με K= 10 θέσεις μνήμης
- Εξισορρόπηση φορτίου μέσω εκ-περιτροπής επιλογή επεξεργαστή (round robin), δηλαδή μία εργασία στην ουρά του πρώτου επεξεργαστή και μια εργασία στην ουρά του άλλου κυκλικά
- Χρόνος προσομοίωσης t = 1000



Εικόνα 4 Ασύμμετρο σύστημα δύο παράλληλων επεξεργαστών και Round Robin ανάθεση

3) Να συμπεριλάβετε εδώ τον κώδικά σας για την παραπάνω προσομοίωση και να παρουσιάσετε τα αποτελέσματα για
   a. Το μέσο χρόνο αναμονής στην ουρά των tasks.
   b. Τον συνολικό-αθροιστικό αριθμό αφίξεων νέων tasks στο σύστημα.
   c. Το συνολικό-αθροιστικό αριθμό αναχωρήσεων-εξυπηρετήσεων των tasks στο σύστημα.
   d. Το πλήθος των εργασιών που έχουν εξυπηρετηθεί από το πρώτο επεξεργαστή, το δεύτερο επεξεργαστή και συνολικά από το σύστημα
   d. Το μέσο συντελεστή χρησιμοποίησης του κάθε επεξεργαστή - ποσοστό utilization.
   e. Το πλήθος των tasks που έχουν γίνει dropped από την πρώτη ουρά, τη δεύτερη ουρά και συνολικά από το σύστημα

ΑΠΑΝΤΗΣΗ:

```
import random

arrivalRate = 5
serviceRate = 6
queueSize = 10
simulationTime = 1000

queue1 = []
queue2 = []
p1BusyUntilTime = float('inf')
p2BusyUntilTime = float('inf')
totalArrivals = 0
totalDepartures = 0
droppedByP1 = 0
droppedByP2 = 0
totalDroppedRequests = 0
totalServedByP1 = 0
totalServedByP2 = 0
totalWaitTime = 0
roundRobinProcessor = 1

currentTime = 0
nextArrival = random.expovariate(arrivalRate)

while currentTime < simulationTime:
currentTime = nextArrival

if p1BusyUntilTime ⩽ currentTime:
p1BusyUntilTime = float('inf')
totalServedByP1 += 1
totalDepartures += 1
if p2BusyUntilTime ⩽ currentTime:
p2BusyUntilTime = float('inf')
totalServedByP2 += 1
totalDepartures += 1

if currentTime == nextArrival:
totalArrivals += 1

if roundRobinProcessor == 1:
if len(queue1) < queueSize:
queue1.append(currentTime)
else:
droppedByP1 += 1

roundRobinProcessor = 2

else:
if len(queue2) < queueSize:
queue2.append(currentTime)
else:
droppedByP2 += 1

roundRobinProcessor = 1

nextArrival = currentTime + random.expovariate(arrivalRate)

if queue1:
if p1BusyUntilTime == float('inf'):
```

```
p1BusyUntilTime = currentTime + random.expovariate(serviceRate)
totalWaitTime += currentTime - queue1.pop(0)

if queue2:
if p2BusyUntilTime == float('inf'):
p2BusyUntilTime = currentTime + random.expovariate(serviceRate)
totalWaitTime += currentTime - queue2.pop(0)
totalDroppedRequests = droppedByP1 + droppedByP2

print(f"Average waiting time in the queue: {round(totalWaitTime / totalDepartures, 2)}")
print(f"Total number of arrivals: {totalArrivals}")
print(f"Total number of departures: {totalDepartures}")
print("Total number of requests served by each processor:")
print(f" Processor 1: {totalServedByP1}")
print(f" Processor 2: {totalServedByP2}")
print("Average utilization of each processor:")
print(f" Processor 1: {round(totalServedByP1 / (simulationTime * serviceRate) * 100, 2)}%")
print(f" Processor 2: {round(totalServedByP2 / (simulationTime * serviceRate) * 100, 2)}%")
print("Number of dropped requests:")
print(f" Processor 1: {droppedByP1}")
print(f" Processor 2: {droppedByP2}")
print(f" Total: {totalDroppedRequests} ({round(totalDroppedRequests / totalArrivals * 100,
2)}%)")
```

```
Average waiting time in the queue: 0.71
Total number of arrivals: 4966
Total number of departures: 4943
Total number of requests served by each processor:
  Processor 1: 2470
  Processor 2: 2473
Average utilization of each processor:
  Processor 1: 41.17%
  Processor 2: 41.22%
Number of dropped requests:
  Processor 1: 13
  Processor 2: 8
  Total: 21 (0.42%)
```

ΠΑΡΑΔΟΣΗ ΕΡΓΑΣΙΑΣ

Παράδοση εργαστηριακής άσκησης πάντα μόνο μέσω eclass και με χρήση του παρόντος απαντητικού φύλλου

Πηγές:

[1] Διαλέξεις των προπτυχιακών και μεταπτυχιακών μαθημάτων *«Απόδοση Παράλληλων Συστημάτων»* και *«Μοντελοποίηση και Προσομοίωση*, ακαδημαϊκά έτη 2010-2012, του Τμ. Πληροφορικής Α.Π.Θ. της καθηγ. Ελένης Καρατζά.

[2] *"Ανάλυση Επίδοσης Υπολογιστικών Συστημάτων: Αναλυτικά μποντέλα, προσομοίωση, μετρήσεις*", Α.Γ. Σταφυλοπάτης, Γ. Σιόλας, Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγραματα και Βοηθήματα, 2015, της Δράσης των Ανοικτών Ακαδημαϊκών Ηλεκτρονικών Συγγραμμάτων - Αποθετήριο Κάλλιπος, Υπερ-σύνδεσμος https://repository.kallipos.gr/handle/11419/6055