# ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

# ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ

# ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

# cslogologo_

# ΕΠΛ425: Τεχνολογίες Διαδικτύου

1η Άσκηση

Ομάδα:

Κωνσταντίνος Διομήδους 934991

Έλενα Κισσοπόδα 915172

**Γενικές πληροφορίες:**

Για τους σκοπούς της δοκιμής χρειάστηκε να δημιουργήσουμε δύο Virtual machines στο amazon web services. Ένα με 1vCPU και ένα με 2vCPU. Ρυθμίσαμε τις μηχανές μας να αποδέχονται το private key μας για να μπορούμε να ενωνόμαστε σε αυτές με ssh. Στη συνέχεια εγκαταστήσαμε τη java(για να μπορεί να τρέξει ο Server μας) και το git( για να μπορούμε να κάνουμε clone το project) σε αυτές τις μηχανές. Ακολούθως ρυθμίσαμε τα virtual machines να αποδέχονται συνδέσεις σε όλα τα ports τους για να μπορεί ο client μας να δημιουργεί σύνδεση με αυτές.

Server:

Ο server μας όταν εκτελεστεί με την εντολή “java server port” ξεκινά να ακούει στη θήρα που του δώσαμε ως παράμετρο για clients. Όταν έρθει κάποιος client ξεκινά ένα καινούργιο νήμα που είναι υπεύθυνο για την εξυπηρέτηση των αιτημάτων από τον συγκεκριμένο χρήστη. Αυτό το νήμα όταν λάβει ένα αίτημα από το Client δημιουργεί ένα τυχαίου μεγέθους response(300-2000KB) και το στέλνει πίσω στο Client. Στο τέλος του response αποστέλλει τη λέξη “finish” για να γνωρίζει ο Client ότι το response έχει τελειώσει. Αυτή η διαδικασία επαναλαμβάνεται έως ότου ο χρήστης αποστείλει την λέξη “Bye”. Όταν τελειώσει με τον client αποθηκεύει στο αρχείο «ServerStatistics» το memory utilization, το CPU load και το throughput.

Client:

Ο client μας όταν εκτελεστεί με την εντολή “java client hostname port N iterations” δημιουργεί Ν νήματα(ένα για κάθε προσομοιωμένο Client). Το κάθε νήμα δημιουργεί σύνδεση με το hostname(το ip του server) και το port(η θύρα που ακούει ο server) που του δώσαμε ως παραμέτρους. Στη συνέχεια στέλνει iterations συνεχόμενα requests στο server. Το κάθε request έχει τη μορφή <Hello clientIpAddress clientPort clientId>. Στη συνέχεια διαβάζει γραμμή-γραμμή το response μέχρι να διαβάσει τη λέξη “finish”. Επίσης μετρά το rtt για κάθε request και στη συνέχεια υπολογίζει το μέσο όρο του και τον αποθηκεύει στο αρχείο «Client Rtt»

Μετά το τέλος των requests αποστέλλει τη λέξη “Bye” στο Server για να ειδοποιήσει το server ότι θα κλείσει τη σύνδεση.

Για την πειραματική αξιολόγηση χρησιμοποιήσαμε 5, 10, 15 και 20 χρήστες. Οι οποίοι έτρεχαν σε υπολογιστές του εργαστηρίου.

Για κάθε ένα από αυτά τα σενάρια περνάμε τις τιμές:

* Latency
* Throughput
* Average CPU load
* Memory Utilization

Χρησιμοποιήσαμε αυτές τις τιμές για να δημιουργήσουμε τις ακόλουθες γραφικές παραστάσεις και να εξάγουμε συμπεράσματα από αυτές.

**Αποτελέσματα:**

Στην πιο πάνω γραφική παράσταση βλέπουμε πως με την χρήση 2vCPU χρειάζεται περισσότερο round trip time/Latency για την ολοκλήρωση ενός request αντί με την χρήση 1vCPU. Ακόμη με την αύξηση των χρηστών βλέπουμε ότι το round trip time/Latency αυξάνεται καθώς ο διακομιστής είναι γεμάτος με requests για να εξυπηρετήσει.

Στην πιο πάνω γραφική παράσταση βλέπουμε πως με την χρήση 2vCPU επιτυγχάνεται πιο μικρό throughput αντί με την χρήση 1vCPU και πως με την αύξηση των χρηστών μειώνονται και των δύο το throughput.

Στην πιο πάνω γραφική παράσταση βλέπουμε πως όσο αυξάνεται το throughput αυξάνεται και το Memory utilization του διακομιστή.

Στην πιο πάνω γραφική παράσταση βλέπουμε πως όσο αυξάνεται το CPU load μειώνεται το throughput γιατί τόσο ο διακομιστής δυσκολεύεται να επεξεργαστεί τις πληροφορίες και κατ’ επέκταση να στείλει τις πληροφορίες στον χρήστη.