Εργασία 4 – Περιβάλλον κατανεμημένης εκτέλεσης με πλήρως διαφανή μετανάστευση

Στάδιο Α: Περιβάλλον εκτέλεσης SimpleScript

Υλοποιήστε ένα περιβάλλον εκτέλεσης για προγράμματα στην γλώσσα SimpleScript, με την παρακάτω σύνταξη:

```
Program = Tag {InstrLine}.
InstrLine = [Label] Instr.
Instr = SET Var VarVal                  |   Var = VarVal

        ADD Var VarValI1 VarValI2       |   Var = VarValI1 + VarValI2
        SUB Var VarValI1 VarValI2       |   Var = VarValI1 - VarValI2
        MUL Var VarValI1 VarValI2       |   Var = VarValI1 * VarValI2
        DIV Var VarValI1 VarValI2       |   Var = VarValI1 / VarValI2
        MOD Var VarValI1 VarValI2       |   Var = VarValI1 % VarValI2

        BGT VarValI1 VarValI2 Label     |   if (VarValI1 > VarValI2) goto Label
        BGE VarValI1 VarValI2 Label     |   if (VarValI1 >= VarValI2) goto Label
        BLT VarValI1 VarValI2 Label     |   if (VarValI1 < VarValI2) goto Label
        BLE VarValI1 VarValI2 Label     |   if (VarValI1 <= VarValI2) goto Label
        BEQ VarValI1 VarValI2 Label     |   if (VarValI1 == VarValI2) goto Label
        BRA Label                       |   goto Label

        PUT VarValS {VarVal}            |   add in tuple space VarValS the tuple with values {VarVal}
        GET VarValS {VarVal}            |   rmv from tuple space VarValS a tuple with fields {VarVal}

        SLP VarValI                     |   sleep(VarValI)
        PRN {VarVal}                    |   print sequence of values {varVal}
        EXT.                                terminate program execution

Tag = #SIMPLESCRIPT
VarVal = VarValI | VarValS.
VarValI = Var | IntVal.
VarValS = Var | StrVal.
Var = '$' {Letter | Digit}.
Label = '#' {Letter | Digit}.
IntVal = [-] Digit {Digit}.
StrVal = " {Letter | Digit} ".
```

Ο κώδικας ενός προγράμματος SimpleScript δίνεται σε ASCII, με κάθε εντολή σε ξεχωριστή γραμμή. Τα τμήματα μιας εντολής χωρίζονται με έναν ή περισσότερους κενούς χαρακτήρες. Υποστηρίζονται μεταβλητές integer και string που δηλώνονται αυτόματα μέσω των εντολών που αναφέρονται σε αυτές. Επίσης, το πρόγραμμα δέχεται ορίσματα που μπορεί να είναι integers ή strings. Τα ορίσματα προσπελάζονται μέσω προκαθορισμένων μεταβλητών `$arg<n>`, όπου `n` ο αριθμός του ορίσματος. Κατά σύμβαση, το `$arg0` είναι ένα string με το όνομα του προγράμματος. Τα υπόλοιπα ορίσματα εξαρτώνται από το πρόγραμμα. Αν ο χρήστης δώσει ασύμβατες τιμές στα ορίσματα του προγράμματος, θα προκληθεί σφάλμα την ώρα της εκτέλεσης.

Αναπτύξτε έναν διερμηνέα SimpleScript που δέχεται το όνομα και τα ορίσματα ενός προγράμματος, και εκτελεί το πρόγραμμα εντολή προς εντολή. Σε περίπτωση συντακτικού λάθους ή ασυμβατότητας τιμών/ορισμάτων μιας εντολής, πρέπει να εκτυπώνεται κατάλληλο μήνυμα και να τερματίζεται η εκτέλεση του προγράμματος. Στην συνέχεια, υλοποιήστε ένα ολοκληρωμένο περιβάλλον που υποστηρίζει την εντολή run <prog> <arg1> … <argN>. Πρέπει να υποστηρίζεται η ταυτόχρονη εκτέλεση προγραμμάτων (πιθανώς μέσα από ξεχωριστά νήματα/διεργασίες). Επίσης, το περιβάλλον πρέπει να προσφέρει την εντολή list για να μπορεί ο χρήστης να δει τις τρέχουσες εκτελέσεις, και την εντολή kill για τον τερματισμό μιας συγκεκριμένης εκτέλεσης.

Το tuple space στο οποίο αναφέρονται οι εντολές `PUT`/`GET` (πρώτο όρισμα, string) πρέπει να δημιουργείται και να καταστρέφεται αυτόματα από το περιβάλλον εκτέλεσης. Επίσης, πρέπει να υποστηρίζονται κοινόχρηστα tuple spaces ανάμεσα σε διαφορετικές εκτελέσεις, αν οι εντολές `PUT`/`GET` αναφέρονται στο ίδιο όνομα.

Στάδιο Β: Κατανεμημένη επικοινωνία

Επεκτείνετε την υλοποίηση σας έτσι ώστε να υποστηρίζονται κοινόχρηστα tuple spaces όχι μόνο στο πλαίσιο του ίδιου περιβάλλοντος εκτέλεσης αλλά και ανάμεσα σε διαφορετικά περιβάλλοντα εκτέλεσης που πιθανώς να βρίσκονται σε ξεχωριστά μηχανήματα.

Για να επιτευχθεί αυτό, πρέπει να υλοποιήσετε κατάλληλους μηχανισμούς/πρωτόκολλα που θα επιτρέπουν στα περιβάλλοντα εκτέλεσης (1) να ανακαλύπτει το ένα το άλλο, (2) να ανταλλάσσουν πληροφορίες για τα tuple spaces που έχουν δημιουργηθεί τοπικά, (3) αν υπάρχει ήδη ένα tuple space με το επιθυμητό όνομα, αυτό να χρησιμοποιείται από όλα τα περιβάλλοντα εκτέλεσης, (4) οι εντολές `PUT`/`GET` να εκτελούνται με διαφανή τρόπο για την εφαρμογή, με την προσπέλαση του tuple space να γίνεται είτε συμβατικά αν είναι τοπικό, είτε μέσω κατάλληλων RPCs αν είναι απομακρυσμένο, και (5) τα tuple spaces να καταστρέφονται αυτόματα όταν δεν υφίσταται πλέον καμία εκτέλεση που να τα χρησιμοποιεί, μέσω κατάλληλου μηχανισμού μέτρησης αναφορών.

Στάδιο Γ: Δυναμική μετανάστευση κώδικα

Υποστηρίξτε την εντολή migrate <id> <IP address> <port> για την μετανάστευση της εκτέλεσης με το αναγνωριστικό id, από το τοπικό σε ένα απομακρυσμένο περιβάλλον εκτέλεσης. Η μετανάστευση πρέπει να γίνεται με διαφανή τρόπο.

Συνοπτικά, η διαδικασία μετανάστευσης αποτελείται από τα εξής στάδια: (1) Αρχικά, το τοπικό περιβάλλον εκτέλεσης σταματά την εκτέλεση του προγράμματος και καταγράφει την κατάσταση του (συμπεριλαμβανομένης και της κατάστασης που σχετίζεται με τις εντολές `PUT`/`GET`). (2) Στην συνέχεια, η κατάσταση μαζί με τον κώδικα του προγράμματος στέλνεται στο απομακρυσμένο περιβάλλον εκτέλεσης. Ως υπηρεσία αξιόπιστης μεταφοράς, χρησιμοποιήστε το λογισμικό network pipes που αναπτύξατε στην 1η εργασία. (3) Τέλος, το απομακρυσμένο περιβάλλον εκτέλεσης, ανακτά/αρχικοποιεί κατάλληλα την κατάσταση του προγράμματος, και συνεχίζει την εκτέλεση του προγράμματος από το σημείο που είχε σταματήσει.

Επίσης, υποστηρίξτε την εντολή shutdown για τον ομαλό τερματισμό του περιβάλλοντος εκτέλεσης, με αυτόματη μετανάστευση των προγραμμάτων που εκτελούνται σε άλλα διαθέσιμα περιβάλλοντα εκτέλεσης. Αν στο περιβάλλον εκτέλεσης τρέχουν ακόμα κάποια προγράμματα και δεν υπάρχει άλλο διαθέσιμο περιβάλλον εκτέλεσης, η εντολή αποτυγχάνει.

Προαιρετικά: Σκεφτείτε πως μπορεί να υποστηριχθεί αυτόματη εξισορρόπηση του φόρτου εργασίας έτσι ώστε τα προγράμματα που εκτελούνται στο σύστημα να μοιράζονται ανάμεσα στα περιβάλλοντα εκτέλεσης που είναι διαθέσιμα ανά πάσα στιγμή.

Προαιρετικά: Εκτέλεση απομακρυσμένου κώδικα

Επεκτείνετε την υλοποίηση σας έτσι ώστε το περιβάλλον εκτέλεσης να μπορεί να εκτελεί προγράμματα ο κώδικας των οποίων βρίσκεται σε έναν απομακρυσμένο εξυπηρετητή. Πιο συγκεκριμένα, αν το αρχείο με τον κώδικα του προγράμματος που δίνει ο χρήστης δεν υπάρχει στον κατάλογο εργασίας, γίνεται αναζήτηση στον εξυπηρετητή. Αν το αρχείο υπάρχει εκεί, η διαδικασία της εκτέλεσης συνεχίζεται κανονικά, με μόνη διαφορά ότι οι εντολές του προγράμματος διαβάζονται από το απομακρυσμένο αρχείο αντί να διαβάζονται από ένα τοπικό αρχείο.

Για την διαφανή πρόσβαση στα περιεχόμενα απομακρυσμένων αρχείων, βασιστείτε στο λογισμικό NFS που αναπτύξατε στην 3η εργασία. Το περιβάλλον εκτέλεσης δεν επιτρέπεται να δημιουργεί τοπικά αντίγραφα των απομακρυσμένων αρχείων, μπορεί όμως να υποθέσει ότι ο κώδικας των προγραμμάτων αλλάζει σπανίως και να θέσει μια αντίστοιχα μεγάλη τιμή για την ισχύ των δεδομένων που διατηρεί ο NFS client στην κρυφή μνήμη.

Χρησιμοποιήστε όποια γλώσσα προγραμματισμού επιθυμείτε. Ο κώδικας σας πρέπει να μεταφράζεται και να εκτελείται κανονικά στο περιβάλλον Linux του εργαστηρίου. Εναλλακτικά, μπορείτε να επιδείξετε την υλοποίηση σε δικά σας laptop που θα πρέπει να συνδέονται στο ενσύρματο δίκτυο του εργαστηρίου. Σιγουρευτείτε ότι η επίδειξη σας δουλεύει σωστά.

Ημερομηνία παράδοσης: **Σάββατο 26 Μαΐου 2018, 22:00.**
Παραδοτέα: (α) κώδικας με οδηγίες εγκατάστασης/χρήσης, (β) παρουσίαση των πρωτοκόλλων και της υλοποίησης.