ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΕΦΟΥΣ ΚΑΙ ΥΠΗΡΕΣΙΕΣ

ΕΡΓΑΣΙΑ 1

ΟΝΟΜΑ: ΣΤΥΛΙΑΝΟΣ

ΕΠΩΝΥΜΟ: ΣΑΓΡΕΔΑΚΗΣ

ΑΜ: 20390302

ΕΡΩΤΗΣΗ 1

Δημιουργούνται 2 Datacenters: Datacenter_0 και Datacenter_1 αυτό προκύπτει από το output.

0.0: Datacenter_0 is starting...
0.0: Datacenter_1 is starting...

Σε κάθε Datacenter δημιουργούνται 2 Hosts αυτό δεν φαίνεται απευθείας από το output αλλά μπορείς να το εντοπίσεις από τα logs του guestAllocator

Datacenter_0.guestAllocator: Vm #0 has been allocated to Host #0
Datacenter_0.guestAllocator: Vm #3 has been allocated to Host #1
Datacenter_1.guestAllocator: Vm #9 has been allocated to Host #1

ΕΡΩΤΗΣΗ 2

Με βάση των πηγαίο κώδικα παρατηρούμε πως ο HOST#0 έχει

PEs (CPU cores): 4
MIPS per
PE: 1000
RAM: 2048 MB
Bandwidth: 10000
Storage: 1,000,000 MB
Scheduler: VmSchedulerTimeShared

Ενώ ο HOST#1 έχει

PEs (CPU cores): 2
MIPS per PE: 1000
RAM: 2048 MB
Bandwidth: 10000
Storage: 1,000,000 MB
Scheduler: VmSchedulerTimeShared

ΕΡΩΤΗΣΗ 3

Με βάση των πηγαίο κώδικα στην σειρά 242,243 παρατηρούμε πως κάθε Datacenter έχει χαρακτηριστικά

Αρχιτεκτονική «arch»
Λειτουργικό σύστημα «os»
Virtual Machine Monitor «vmm»
Λίστα από Host «hostList»
Χρονική ζώνη του datacenter «time_zone»
Κόστη υπολογιστικών πόρων «cost, costPerMem, costPerStorage, costPerBw»

ΕΡΩΤΗΣΗ 4

Σύμφωνα με τη γραμμή createVM(brokerId, 20); στον κώδικα, δημιουργούνται 20 VMs. Αυτό μπορούμε να το επιβεβαιώσουμε και από το output όπου ο Broker προσπαθεί να δημιουργήσει VM #0 έως VM #19.Επίσης τα χαρακτηριστικά που έχουν είναι τα ίδια σε όλες εκτός από το ID:

Η τιμή του (το i), το user id του, η υπολογιστική ισχύ του (MIPS), ένας επεξεργαστής αν VM(PEs), η μνήμη (RAM), η δικτυακή ταχύτητα (Bandwidth), το μέγεθος του δίσκου (storage), Virtual Machine Monitor (VMM) και η δρομολόγηση (Cloudlet Scheduler).

ΕΡΩΤΗΣΗ 5

Αυτό εξαρτάται από τους πόρους που παρέχει ο Host και από τους πόρους που ζητάει κάθε VM δηλαδή ο Host0 μπορεί να δεχτεί μέχρι 4VMs από άποψη CPU και RAM ενώ ο HOST1 μπορεί να δεχτεί μέχρι 2 από άποψη CPU και μέχρι 4 από άποψη RAM. Άρα κάθε Datacenter μπορεί να φιλοξενήσει μέχρι 6VMs. Επίσης μπορούμε να το δούμε και από το output. Ο αλγόριθμος που μπορούμε να το εντοπίσουμε αυτό είναι από την κλάση org.cloudbus.cloudsim.VmAllocationPolicySimple η οποία εφαρμόζει fisrt-fit allocation δηλαδή ελέγχει κάθε HOST με την σειρά. Αν καταφέρει και βρει HOST με αρκετούς διαθέσιμους πόρους αναθέτει εκεί την VM αν εκείνη την στιγμή κανένας HOST δεν έχει αρκετούς διαθέσιμους πόρους τότε η VM απορρίπτεται. Αυτό μπορούμε να το επιβεβαιώσουμε και από το output το οποίο στην αρχή κάνει ανάθεση σε HOST και μετά αποτυγχάνει.

ΕΡΩΤΗΣΗ 6

Από το output παρατηρούμε πως 6VMs ανατέθηκαν στο Datacenter_0 άλλες 6 στο 1 ενώ οι υπόλοιπες απέτυχαν. Ο λόγος που γίνεται αυτό είναι διότι οι HOST έχουν περιορισμένους πόρους δηλαδή κάθε HOST έχει συγκεκριμένο αριθμό επεξεργαστών και περιορισμένη RAM. Στην περίπτωση που κάθε VM είχε 2 PEs των 250 MIPS και 256 MB RAM αυτό σημαίνει ότι μπορεί να φιλοξενήσει λιγότερα VMs δηλαδή στο HOST0 από 4 θα μπορεί 2 και στον HOST1 από 2 μόνο 1. Άρα κάθε Datacenter θα μπορούσε να εκτελέσει μόνο 3 VMs αντί για 6.

ΕΡΩΤΗΣΗ 7

Από τον κώδικα main δημιουργούνται 40 Cloudlets και αποθηκεύονται στην λίστα cloudletList
«*cloudletList* = *createCloudlet*(brokerId,40); // creating 40 cloudlets»

Τα χαρακτηριστικά τους είναι όλα τα ίδια με εξαίρεση το id τους και είναι τα εξής i, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel.

ΕΡΩΡΗΣΗ 8

Από τον κώδικα broker.submitCloudletList(cloudletList); Ο DataCenterBroker αναλαμβάνει να κατανεμήσει τα Clouddlets στις διαθέσιμες VMs Ο DatacenterBroker χρησιμοποιεί πολιτική first-fit round-robin δηλαδή αναθέτει τα Cloudlets κυκλικά στις διαθέσιμες VMs. Αυτή η ανάθεση γίνεται μόνο σε VMs που έχουν εκκινήσει επιτυχώς. Επίσης δεν υπάρχει load balancing — μόνο εναλλαγή ανά διαθέσιμη VM. Αυτό μπορούμε να το δούμε και από το output. Επίσης θα παρατηρήσουμε ότι και τα 40 εκτελέστηκαν επιτυχώς όπου ανατέθηκαν στις 12 διαθέσιμες VMs όπου το καθένα από αυτά εκτέλεσε περίπου 3-4 Cloudlets.

ΕΡΩΤΗΣΗ 9

A)

new CloudletSchedulerTimeShared()

org.cloudbus.cloudsim > CloudletSchedulerTimeShared.java

Η VM χρησιμοποιεί Time Sharing πολιτική για την εκτέλεση Cloudlets όλοι οι Cloudlets που ανατίθενται σε μία VM εκτελούνται ταυτόχρονα με κατανομή χρόνου οι πόροι μοιράζονται μεταξύ των Cloudlets.

B) new VmSchedulerTimeShared(peList)

Ο Host χρησιμοποιεί Time Sharing πολιτική για την εκτέλεση των VMs του αντί οι VMs να καταλαμβάνουν αποκλειστικά PEs μοιράζονται τους PEs μέσω χρόνο.

ΕΡΩΤΗΣΗ 10

Ο πίνακας εμφανίζει για κάθε Cloudlet:

- Ποια VM και ποιο Datacenter το εκτέλεσε
- Τους χρόνους εκκίνησης και ολοκλήρωσης
- Τη διάρκεια εκτέλεσης

Αυτά επηρεάζονται από τις πολιτικές Time Sharing, τόσο:

- CloudletSchedulerTimeShared (στο επίπεδο VM)
- VmSchedulerTimeShared (στο επίπεδο Host)

Αυτό εξηγεί γιατί ορισμένα Cloudlets έχουν παρόμοιους χρόνους εκκίνησης αλλά διαφορετικούς χρόνους εκτέλεσης.

ΕΡΩΤΗΣΗ Α

Στο ερώτημα αυτό μελετήσαμε την συμπεριφορά και την απόδοση του CloudSim χρησιμοποιώντας διαφορετικές πολιτικές δρομολόγησης. Πιο συγκεκριμένα χρησιμοποιήσαμε τις πολιτικές.

(α) Δρομολόγηση Cloudlets στα VMs με τις πολιτικές:

- CloudletSchedulerSpaceShared
- CloudletSchedulerTimeShared

(β) Δρομολόγηση VMs στους Hosts με τις πολιτικές:

- VmSchedulerSpaceShared
- VmSchedulerTimeShared
- VmSchedulerTimeSharedOverSubscription

Οι διαφορές στους πρώτους δρομολογητές είναι

Για το CloudletSchedulerSpaceShared:

- Κάθε Cloudlet εκτελείται αποκλειστικά πάνω σε ένα VM.
- Ο επόμενος Cloudlet ξεκινά μόνο όταν ολοκληρωθεί ο προηγούμενος.
- Πλεονεκτήματα: προβλέψιμος χρόνος εκτέλεσης.
- Μειονεκτήματα: μειωμένη αξιοποίηση πόρων και μεγαλύτερος συνολικός χρόνος εκτέλεσης.

Για το CloudletSchedulerTimeShared:

- Όλοι οι Cloudlets εκτελούνται ταυτόχρονα, μοιράζοντας τον χρόνο CPU.
- Ξεκινάνε όλοι σχεδόν ταυτόχρονα και ολοκληρώνονται σε κυματισμούς.
- Πλεονεκτήματα: καλύτερη απόδοση, μικρότερος χρόνος ολοκλήρωσης (σε σύγκριση με SpaceShared).
- Μειονεκτήματα: ελαφρώς πιο πολύπλοκη διαχείριση.

Συμπέρασμα: Το TimeShared είναι πιο αποδοτικό για μαζική και παράλληλη εκτέλεση Cloudlets.

Οι διαφορές στους δεύτερους δρομολογητές

Για VmSchedulerSpaceShared:

- Κάθε VM δεσμεύει τους φυσικούς επεξεργαστές του Host αποκλειστικά.
- Οι VMs εκτελούνται σειριακά όταν δεν υπάρχουν διαθέσιμοι πόροι.

Για VmSchedulerTimeShared:

- Τα VMs μοιράζονται τους PEs με χρονομερισμό.
- Επιτρέπει την παράλληλη λειτουργία περισσότερων VMs.

Για VmSchedulerTimeSharedOverSubscription:

- Επιτρέπει oversubscription: περισσότερα VMs από τους διαθέσιμους PEs.
- Αν και οι πόροι δεν επαρκούν πλήρως, τα VMs εκτελούνται με μοιρασμένο υπολογιστικό χρόνο.
- Στο output παρατηρήσαμε ότι πολλά Cloudlets εκτελέστηκαν ταυτόχρονα, σε πολύ σύντομο χρόνο.
- Δεν υπήρχαν σφάλματα ή καθυστερήσεις — επιτυχής εκτέλεση όλων των Cloudlets.

Συμπέρασμα: Το TimeSharedOverSubscription επιτυγχάνει τη μέγιστη αξιοποίηση των υπολογιστικών πόρων, επιτρέποντας την εκτέλεση περισσότερων VMs ακόμη και όταν οι διαθέσιμοι PEs δεν επαρκούν. Είναι ιδανικό για περιβάλλοντα με υψηλό φόρτο και δυναμικές απαιτήσεις.

Γενικό Συμπέρασμα

Η πολιτική TimeSharedOverSubscription σε συνδυασμό με CloudletSchedulerTimeShared προσφέρει:

- Την καλύτερη απόδοση, ειδικά σε σενάρια με μεγάλο αριθμό Cloudlets.
- Μέγιστη αξιοποίηση των διαθέσιμων Hosts και VMs.
- Γρήγορη και ομαλή εκτέλεση με μειωμένο συνολικό χρόνο.

Αντίθετα, οι πολιτικές SpaceShared είναι κατάλληλες μόνο όταν απαιτείται εγγυημένος καταμερισμός πόρων χωρίς επιμερισμό.

ΕΡΩΤΗΜΑ Β

Α) Με την αύξηση μόνο των χαρακτηριστικών των HOST παρατηρούμε ότι όλες οι VMs εκτελέστηκαν με επιτυχία. Επίσης δεν υπήρξε oversubscription και οι Cloudlets διανεμήθηκαν σωστά. Αποτέλεσμα σε αυτά είναι πως ο χρόνος εκτέλεσης ήταν γρήγορος καθώς οι ισχυρότεροι hosts εξυπηρετούσαν περισσότερα VMs με ευκολία.

Β) Αυτό έχει σαν αποτέλεσμα κάθε PE να μπορεί να υποστηρίξει περισσότερες VMs όπου όλες εκτελέστηκαν χωρίς λάθος όπως επίσης οι Cloudlets κατανεμήθηκαν ισομερώς στα διαθέσιμα VMs με αποτέλεσμα να επιτρέπει μεγαλύτερη παράλληλη εκτέλεση βελτιώνοντας τη χωρητικότητα χωρίς ανάγκη για περισσότερα μηχανήματα.

Γ) Η διαφοροποίηση αυτή έφερε σαν αποτέλεσμα την σωστή κατανομή VMs όπως επίσης το φορίο ανα HOST είναι πιο κατανεμημένο ενώ ο χρόνος εκτέλεσης παρέμεινε σχεδόν ίδιος και τέλος υπάρχει μεγαλύτερη αξιοπιστία σε απομονωμένες εκτελέσεις.