ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΕΦΟΥΣ ΚΑΙ ΥΠΗΡΕΣΙΕΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΙΑ #1

ΣΕΝΑΡΙΑ ΣΤΟ CLOUDSIM

ΣΤΟΙΧΕΙΑ ΕΡΓΑΣΙΑΣ

ΥΠΕΥΘΥΝΟΙ ΘΕΩΡΙΑΣ: ΚΑΛΛΕΡΓΗΣ ΔΗΜΗΤΡΙΟΣ – ΜΑΜΑΛΗΣ ΒΑΣΙΛΕΙΟΣ
ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ : 16/6/2024
ΠΡΟΘΕΣΜΙΑ ΥΠΟΒΟΛΗΣ : 16/06/2024

ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΕΦΟΥΣ ΚΑΙ ΥΠΗΡΕΣΙΕΣ

ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ

ΦΩΤΟΓΡΑΦΙΑ ΦΟΙΤΗΤΗ:

ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΑΘΑΝΑΣΙΟΥ ΒΑΣΙΛΕΙΟΣ ΕΥΑΓΓΕΛΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ : 19390005
ΕΞΑΜΗΝΟ ΦΟΙΤΗΤΗ : 10
ΚΑΤΑΣΤΑΣΗ ΦΟΙΤΗΤΗ : ΠΡΟΠΤΥΧΙΑΚΟ
ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ : ΠΑΔΑ

ΠΕΡΙΕΧΟΜΕΝΑ

Θεωρητικό Μέρος

1. Πόσα Datacenters δημιουργούνται; Πόσοι Hosts δημιουργούνται σε κάθε Datacenter;

Στην προσομοίωση του παραδείγματος CloudSimExample6 δημιουργούνται:

1. **Datacenters (Κέντρα Δεδομένων)**

2 Datacenters (Datacenter_0, Datacenter_1), με entity_id=2 και entity_id=3 αντίστοιχα

CloudSimExample6/main(String[] args)

```
// Second step: Create Datacenters
                    //Datacenters are the resource providers in CloudSim. We need at
list one of them to run a CloudSim simulation
                    @SuppressWarnings("unused")
                    Datacenter datacenter0 = createDatacenter("Datacenter_0");
                    @SuppressWarnings("unused")
                    Datacenter datacenter1 = createDatacenter("Datacenter_1");
```

2. **Hosts (Φυσικές μηχανές φιλοξενείας)**

2 Hosts αντίστοιχα σε κάθε Datacenter, με entity_id=0 και entity_id=1 αντίστοιχα

CloudSimExample6/createDatacenter(String name) : Datacenter

```
            hostList.add(
                  new Host(
                        hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                        storage,
                        peList1,
                        new VmSchedulerTimeShared(peList1)
                  )
            ); // This is our first machine

            hostId++;

            hostList.add(
                  new Host(
                        hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                        storage,
                        peList2,
```

```
                        new VmSchedulerTimeShared(peList2)
                    )
            ); // Second machine
```

2. Πόσους επεξεργαστές (PEs) και τι άλλα χαρακτηριστικά έχει κάθε Host (mips, μνήμη κλπ.);

Ο κάθε Host διαθέτει τα ακόλουθα χαρακτηριστικά:

1. **Στοιχεία επεξεργασίας (PEs)**

   Ο Host με entity_id=0 διαθέτει 4 στοιχεία επεξεργασίας, ενώ ο Host με entity_id=1 διαθέτει 2 στοιχεία επεξεργασίας.

2. **Απόδοση στοιχείου επεξεργασίας (MIPS)**

   Η απόδοση ενός στοιχείου επεξεργασίας του κάθε Host είναι 1000 MIPS, που σημαίνει ότι το κάθε στοιχείο επεξεργασίας μπορεί να εκτελέσει 1000 * 1000000 = 1000000000 εντολές το δευτερόλεπτο.

3. **Μνήμη (RAM)**

   Και οι δύο Host διαθέτουν από 2048 MB μνήμη.

4. **Αποθηκευτικός χώρος (Storage)**

   Και οι δύο Host διαθέτουν από 1000000 MB (1 TB) αποθηκευτικό χώρο.

5. **Εύρος ζώνης δικτύου (BW)**

   Και οι δύο Host διαθέτουν από 10000 MB/s εύρος ζώνης δικτύου.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
// Here are the steps needed to create a PowerDatacenter:
            // 1. We need to create a list to store one or more
            //    Machines
            List<Host> hostList = new ArrayList<Host>();

            // 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should
            //    create a list to store these PEs before creating
            //    a Machine.
            List<Pe> peList1 = new ArrayList<Pe>();

            int mips = 1000;

            // 3. Create PEs and add these into the list.
            //for a quad-core machine, a list of 4 PEs is required:
```

```
            peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating
            peList1.add(new Pe(1, new PeProvisionerSimple(mips)));
            peList1.add(new Pe(2, new PeProvisionerSimple(mips)));
            peList1.add(new Pe(3, new PeProvisionerSimple(mips)));

            //Another list, for a dual-core machine
            List<Pe> peList2 = new ArrayList<Pe>();

            peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
            peList2.add(new Pe(1, new PeProvisionerSimple(mips)));

            //4. Create Hosts with its id and list of PEs and add them to the list of
machines
            int hostId=0;
            int ram = 2048; //host memory (MB)
            long storage = 1000000; //host storage
            int bw = 10000;
```

3. Τι άλλα χαρακτηριστικά έχει το κάθε Datacenter; Σε ποια κλάση αναπαρίστανται;

Τα χαρακτηριστικά που έχει το κάθε Datacenter είναι τα εξής:

1. **Σύστημα αρχιτεκτονικής (arch)**

   Το κάθε Datacenter χρησιμοποιεί επεξεργαστές με βάση το x86 για σύστημα αρχιτεκτονικής του υλικού.

2. **Λειτουργικό σύστημα (OS)**

   Το λειτουργικό σύστημα που είναι εγκατεστημένο στις μηχανές του κάθε Datacenter είναι το Linux.

3. **Διαχείριση εικονικής μηχανής (VMM)**

   Το λογισμικό που διαχειρίζεται τις εικονές μηχανές στους υπολογιστές του κάθε Datacenter είναι το Xen.

4. **Ζώνη ώρας (time_zone)**

   Η ζώνη ώρας στην οποία βρίσκεται το κάθε Datacenter είναι 10 (GMT+ 10).

5. **Παράμετροι κόστους**

   - **Κόστος επεξεργασίας (cost)**

Το κόστος που δαπανάται για τους πόρους επεξεργασίας του κάθε Datacenter είναι 3 G$/Pe.

- **Κόστος μνήμης (costPerMem)**

  Το κόστος που δαπανάται για τους πόρους μνήμης του κάθε Datacenter είναι 0.05 G$/Pe.

- **Κόστος χώρου (costPerStorage)**

  Το κόστος που δαπανάται για τους πόρους αποθηκευτικού χώρου του κάθε Datacenter είναι
  0.1 G$/Pe.

- **Κόστος εύρους ζώνης (costPerBw)**

  Το κόστος που δαπανάται για τους πόρους του εύρους ζώνης του δικτύου του κάθε Datacenter είναι 0.1 G$/Pe.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
// 5. Create a DatacenterCharacteristics object that stores the
            //    properties of a data center: architecture, OS, list of
            //    Machines, allocation policy: time- or space-shared, time zone
            //    and its price (G$/Pe time unit).
            String arch = "x86";       // system architecture
            String os = "Linux";       // operating system
            String vmm = "Xen";
            double time_zone = 10.0;   // time zone this resource located
            double cost = 3.0;         // the cost of using processing in this resource
            double costPerMem = 0.05;   // the cost of using memory in this resource
            double costPerStorage = 0.1;// the cost of using storage in this resource
            double costPerBw = 0.1;     // the cost of using bw in this resource
```

Τα Datacenters αναπαρίστανται στην κλάση **Datacenter.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim.**

CloudSimExample6/createDatacenter(String name) : Datacenter

```
// 6. Finally, we need to create a PowerDatacenter object.
            Datacenter datacenter = null;
            try {
                  datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
            } catch (Exception e) {
                  e.printStackTrace();
            }
```

4. Πόσες VM δημιουργούνται; Τι χαρακτηριστικά έχει η κάθε μία;

Στην προσομοίωση του παραδείγματος CloudSimExample6 δημιουργούνται:

1. **VM (Εικονικές μηχανές)**

   20 VMs με entity_id=0 εώς entity_id=19.

```
CloudSimExample6/main(String[] args)
```

```
//Fourth step: Create VMs and Cloudlets and send them to broker
                    vmlist = createVM(brokerId,20); //creating 20 vms
```

Η κάθε εικονική μηχανή διαθέτει τα εξής χαρακτηριστικά:

1. **Μέγεθος ειδώλου (image size)**

   Κάθε εικονική μηχανή έχει μέγεθος 10000 MB (10 GB).

2. **Μνήμη (RAM)**

   Κάθε εικονική μηχανή διαθέτει 512 MB μνήμη RAM.

3. **Απόδοση στοιχείου επεξεργασίας (MIPS)**

   Η απόδοση ενός στοιχείου επεξεργασίας κάθε εικονικής μηχανής είναι 1000 MIPS, που σημαίνει ότι ένα στοιχείο επεξεργασίας μπορεί να εκτελέσει 1000 * 1000000 = 1000000000 εντολές το δευτερόλεπτο.

4. **Εύρος ζώνης δικτύου (bw)**

   Το εύρος ζώνης δικτύου που διαθέτει η κάθε εικονική μηχανή είναι 1000 MB/s.

5. **Αριθμός στοιχείων επεξεργασίας (PEs)**

   Κάθε εικονική μηχανή διαθέτει και από 1 στοιχείο επεξεργασίας.

6. **Διαχειριστής εικονικής μηχανής (VMM)**

   Το λογισμικό που διαχειρίζεται την κάθε εικονική μηχανή είναι το Xen.

CloudSimExample6/createVM(int userId, int vms) : List<Vm>

```
//VM Parameters
            long size = 10000; //image size (MB)
            int ram = 512; //vm memory (MB)
            int mips = 1000;
            long bw = 1000;
            int pesNumber = 1; //number of cpus
            String vmm = "Xen"; //VMM name
```

5. Από ποιες παραμέτρους καθορίζεται το πόσες VM μπορεί να φιλοξενήσει κάθε Host; Και πως γίνεται (με τι αλγόριθμο) η προσπάθεια ανάθεσής τους στους διαθέσιμους Hosts (πως αποφασίζεται δηλαδή αν και σε ποιον Host θα τρέξει η κάθε VM);

Οι παράμετροι που καθορίζουν το πόσες VM μπορεί να φιλοξενήσει κάθε Host είναι:

1. **Απόδοση επεξεργαστή (MIPS)**

   Κάθε εικονική μηχανή απαιτεί έναν ορισμένο αριθμό PE με συγκεκριμένο MIPS (εκατομμύρια εντολές το δευτερόλεπτο). Ο συνολικός αριθμός των διαθέσιμων PEs και η απόδοση τους (MIPS) στον Host καθορίζει πόσες εικονικές μηχανές (VMs) μπορεί να υποστηρίξει.

2. **Μνήμη (RAM)**

   Κάθε VM απαιτεί ορισμένη μνήμη RAM. Η συνολική διαθέσιμη μνήμη RAM στον Host μαζί με τη μνήμη που διαθέτουν τα VM που φιλοξενεί, καθορίζει τον αριθμό των πρόσθετων VM που μπορεί ο Host να φιλοξενήσει.

CloudSimExample6/Output

```
[VmScheduler.vmCreate] Allocation of VM #6 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #6 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #7 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #7 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #8 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #8 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #9 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #9 to Host #1 failed by MIPS
```

1. **Χωρητικότητα αποθήκευσης (Storage)**

Κάθε VM απαιτεί ένα ορισμένο χώρο αποθήκευσης. Ο συνολικός διαθέσιμος αποθηκευτικός χώρος στον Host μαζί με τον αποθηκευτικό χώρο που διαθέτουν τα VM που φιλοξενεί, καθορίζει τον αριθμό των πρόσθετων VM που μπορεί ο Host να φιλοξενήσει.

2. **Εύρος ζώνης δικτύου (BW)**

Κάθε VM μπορεί να απαιτεί ένα συγκεκριμένο εύρος ζώνης δικτύου.Το συνολικό διαθέσιμο εύρος ζώνης και το εύρος ζώνης που διατίθεται στα υπάρχοντα VM μπορεί να περιορίσει τον αριθμό των πρόσθετων VM.

Η προσπάθεια ανάθεσης των VMs στους Hosts διαχειρίζεται από την κλάση **VmAllocationPolicySimple.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**. Ο Host με τα λιγότερα στοιχεία επεξεργασίας (PEs) που είναι σε χρήση από άλλα VMs, θα επιλεχθεί για να φιλοξενήσει το επόμενο VM που αιτείται κατά αύξουσα σειρά entity_id. Η πολιτική κατανομής των VM δεν είναι η βέλτιστη δυνατή και γι' αυτό τον λόγο ονομάζεται **Worst-Fit** πολιτική.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
// 6. Finally, we need to create a PowerDatacenter object.
            Datacenter datacenter = null;
            try {
                  datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
            } catch (Exception e) {
                  e.printStackTrace();
            }
```

6. Πόσες VMs θα τρέξουν τελικά και σε ποιον Host η κάθε μία; Για ποιο λόγο κάποιες VM δεν θα ανατεθούν σε κανέναν Host; Πώς θα άλλαζαν τα παραπάνω (και γιατί) αν η κάθε VM δημιουργείτο αρχικά με 2 PEs των 250 MIPS και 256 MB RAM;

Συνολικά, δημιουργούνται 20 VMs, όπως απαντήθηκε στο ερώτημα 4, και η κατάσταση ανάθεσης τους στους Hosts των Datacenters (για το πλήθος των Datacenters και των Hosts σε κάθε Datacenter που δημιουργούνται κατά την προσομοίωση, βλ. ερώτημα 1) είναι η εξής:

1. **Datacenter_0 (#2)**

   - **Host #0**

     VM #0, #1, #2, #4 (Συνολικά 4 VMs τρέχουν σ' αυτό τον Host).

   - **Host #1**

     VM #3, #5 (Συνολικά 2 VMs τρέχουν σ' αυτό τον Host).

CloudSimExample6/Output

```
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
```

1. **Datacenter_1 (#3)**

   - **Host #0**

     VM #6, #7, #8, #10 (Συνολικά 4 VMs τρέχουν σ' αυτό τον Host).

   - **Host #1**

     VM #9, #11 (Συνολικά 2 VMs τρέχουν σ' αυτό τον Host)

CloudSimExample6/Output

```
0.2: Broker: VM #6 has been created in Datacenter #3, Host #0
0.2: Broker: VM #7 has been created in Datacenter #3, Host #0
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #1
0.2: Broker: VM #10 has been created in Datacenter #3, Host #0
0.2: Broker: VM #11 has been created in Datacenter #3, Host #1
```

Οι 20 εικονικές μηχανές διαθέτουν τα ίδια χαρακτηριστικά (βλ. ερώτημα 4).

Οι Hosts επίσης διαθέτουν τα ίδια χαρακτηριστικά με τη μόνη διαφορά στο πλήθος των επεξεργαστών PEs (βλ. ερώτημα 2).

Ο λόγοι που κάποιες VM δεν θα ανατεθούν σε κανέναν Host (για παραμέτρους που καθορίζουν το πόσες VM μπορεί να φιλοξενήσει ένας Host, βλ. ερώτημα 5) είναι οι εξής:

1. **Απόδοση επεξεργαστή (MIPS)**

   Ο Host #1 μπορεί να φιλοξενήσει μέχρι 2 VMs λόγω απόδοσης του επεξεργαστή (MIPS). Ο Host #1 διαθέτει 2 επεξεργαστές PEs, ενώ το κάθε VM διαθέτει από 1. Δεδομένου, ότι η απόδοση των επεξεργαστών είναι ίδια (1000 MIPS), δεν καθιστά εφικτό ο Host #1 να φιλοξενήσει παραπάνω από 2 VMs.

2. **Μνήμη (RAM)**

   Ο Host #0 μπορεί να φιλοξενήσει μέχρι 4 VMs λόγω μνήμης RAM. Συγκεκριμένα, ο Host #0 διαθέτει 2048 MB μνήμη, ενώ, κάθε VM από 512 MB μνήμη (512 * 4 = 2048). Συνεπώς, η φιλοξενία ενός 5^{ου} VM δεν καθιστά εφικτή, καθώς με φιλοξενία τεσσάρων VM, η διαθέσιμη μνήμη του Host #0 έχει εξαντληθεί.

CloudSimExample6/Output

```
[VmScheduler.vmCreate] Allocation of VM #6 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #6 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #7 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #7 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #8 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #8 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #9 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #9 to Host #1 failed by MIPS
```

Όπως απαντήθηκε στο ερώτημα 5, η πολιτική ανάθεσης των VMs στους Hosts είναι η **Worst-Fit** πολιτική. Οι αλλαγές που προέκυψαν στις παραμέτρους των VM είναι ως προς την απόδοση του επεξεργαστή που μειώθηκε από τον 1 επεξεργαστή των 1000 MIPS στους 2 επεξεργαστές των 250 MIPS. Επίσης, μειώθηκε και η μνήμη RAM από 512 MB στα 256 MB. Συνεπώς, με τη μείωση των παραμέτρων αυτών, οι Hosts μπορούν να φιλοξενήσουν περισσότερα VMs και η νέα κατάσταση έχει ως εξής:

1. **Datacenter_0 (#2)**

   - **Host #0**

     VM #0, #1, #3, #5, #7, #9, #10, #11 (Συνολικά 8 VMs τρέχουν σ' αυτό τον Host).

- **Host #1**

  VM #2, #4, #6, #8 (Συνολικά 4 VMs τρέχουν σ' αυτό τον Host).

```
CloudSimExample6/Output

0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #1
0.1: Broker: VM #3 has been created in Datacenter #2, Host #0
0.1: Broker: VM #4 has been created in Datacenter #2, Host #1
0.1: Broker: VM #5 has been created in Datacenter #2, Host #0
0.1: Broker: VM #6 has been created in Datacenter #2, Host #1
0.1: Broker: VM #7 has been created in Datacenter #2, Host #0
0.1: Broker: VM #8 has been created in Datacenter #2, Host #1
0.1: Broker: VM #9 has been created in Datacenter #2, Host #0
0.1: Broker: VM #10 has been created in Datacenter #2, Host #0
0.1: Broker: VM #11 has been created in Datacenter #2, Host #0
```

2. **Datacenter_1 (#3)**

- **Host #0**

  VM #12, #13, #15, #17, #19 (Συνολικά 5 VMs τρέχουν σ' αυτό τον Host)

- **Host #1**

  VM #14, #16, #18 (Συνολικά 3 VMs τρέχουν σ' αυτό τον Host)

```
CloudSimExample6/Output

0.2: Broker: VM #12 has been created in Datacenter #3, Host #0
0.2: Broker: VM #13 has been created in Datacenter #3, Host #0
0.2: Broker: VM #14 has been created in Datacenter #3, Host #1
0.2: Broker: VM #15 has been created in Datacenter #3, Host #0
0.2: Broker: VM #16 has been created in Datacenter #3, Host #1
0.2: Broker: VM #17 has been created in Datacenter #3, Host #0
0.2: Broker: VM #18 has been created in Datacenter #3, Host #1
0.2: Broker: VM #19 has been created in Datacenter #3, Host #0
```

Όπως παρατηρούμε και στο output της προσομοίωσης, και τα 20 VMs δημιουργούνται και κατανέμονται στους Hosts.

Ο Host #0 με 4 στοιχεία επεξεργασίας των 4000 MIPS συνολικής απόδοσης, μπορεί να φιλοξενήσει το πολύ 8 VMs με 2 στοιχεία επεξεργασίας των 500 MIPS συνολικής απόδοσης.

Ο Host #1 με 2 στοιχεία επεξεργασίας των 2000 MIPS συνολικής απόδοσης, μπορεί να φιλοξενήσει το πολύ 4 VMs με με 2 στοιχεία επεξεργασίας των 500 MIPS συνολικής απόδοσης.

Η διαθέσιμη μνήμη εξαντλείται στην 1η περίπτωση όπως και ίσχυε στις αρχικές παραμέτρους των VMs.

7. Πόσα Cloudlets δημιουργούνται; Με τι χαρακτηριστικά το κάθε ένα;

Στην προσομοίωση του παραδείγματος CloudSimExample6 δημιουργούνται:

1. **Cloudlets (Διεργασίες)**

   40 Cloudlets με entity_id=0 εώς entity_id=39

CloudSimExample6/main(String[] args)

```
//Fourth step: Create VMs and Cloudlets and send them to broker
                    vmlist = createVM(brokerId,20); //creating 20 vms
                    cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets
```

Τα Cloudlets διαθέτουν τα εξής χαρακτηριστικά:

1. **Μέγεθος (length)**

   Το μέγεθος του Cloudlet καθορίζεται από τον αριθμό των εντολών που πρέπει να εκτελέσει. Κάθε Cloudlet απαιτεί 1000 MI (Millions Instructions) για να ολοκληρωθεί η εκτέλεση του. Η απόδοση του επεξεργαστή (μετριέται σε MIPS) και το μέγεθος του Cloudlet καθορίζουν τον χρόνο εκτέλεσης του Cloudlet στο VM.

2. **Μέγεθος αρχείου εισόδου (fileSize)**

   Το μέγεθος αρχείου εισόδου που χρειάζεται το κάθε Cloudlet για να ξεκινήσει την εκτέλεση του είναι 300 bytes.

3. **Μέγεθος αρχείου εξόδου (outputSize)**

   Το μέγεθος αρχείου εξόδου που θα παράγει το κάθε Cloudlet κατά την ολοκήρωση της εκτέλεσης του είναι 300 bytes.

4. **Αριθμός στοιχείων επεξεργασίας (PEs)**

   Κάθε Cloudlet απαιτεί 1 PE για να εκτελεστεί.

CloudSimExample6/createCloudlet(int userId, int cloudlets) : List<Cloudlet>

```
//cloudlet parameters
            long length = 1000;
            long fileSize = 300;
            long outputSize = 300;
            int pesNumber = 1;
```

8. Πως αποφασίζεται για κάθε Cloudlet σε ποια VM θα τρέξει (με τι αλγόριθμο γίνεται δηλαδή η κατανομή των Cloudlets στις διαθέσιμες VMs); Πόσα Cloudlets τελικά θα εκτελεστούν (τόσο συνολικά σε όλες τις VM όσο και σε κάθε VM ξεχωριστά);

Το κάθε Cloudlet εκχωρείται στην πρώτη εικονική μηχανή (VM) που ικανοποιεί τις υπολογιστικές απαιτήσεις του. Όλες οι VM έχουν τα ίδια χαρακτηριστικά και ικανοποιούν τις απαιτήσεις του κάθε Cloudlet που έχουν επίσης τα ίδια χαρακτηριστικά (για χαρακτηριστικά VM βλ. ερώτημα 4, για χαρακτηριστικά Cloudlets βλ. ερώτημα 7). Η πολιτική αυτή ονομάζεται **First-Fit**.

CloudSimExample6/Output

```
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
0.2: Broker: Sending cloudlet 2 to VM #2
0.2: Broker: Sending cloudlet 3 to VM #3
0.2: Broker: Sending cloudlet 4 to VM #4
0.2: Broker: Sending cloudlet 5 to VM #5
0.2: Broker: Sending cloudlet 6 to VM #6
0.2: Broker: Sending cloudlet 7 to VM #7
0.2: Broker: Sending cloudlet 8 to VM #8
0.2: Broker: Sending cloudlet 9 to VM #9
0.2: Broker: Sending cloudlet 10 to VM #10
0.2: Broker: Sending cloudlet 11 to VM #11
0.2: Broker: Sending cloudlet 12 to VM #0
0.2: Broker: Sending cloudlet 13 to VM #1
0.2: Broker: Sending cloudlet 14 to VM #2
0.2: Broker: Sending cloudlet 15 to VM #3
0.2: Broker: Sending cloudlet 16 to VM #4
0.2: Broker: Sending cloudlet 17 to VM #5
0.2: Broker: Sending cloudlet 18 to VM #6
0.2: Broker: Sending cloudlet 19 to VM #7
0.2: Broker: Sending cloudlet 20 to VM #8
0.2: Broker: Sending cloudlet 21 to VM #9
0.2: Broker: Sending cloudlet 22 to VM #10
0.2: Broker: Sending cloudlet 23 to VM #11
0.2: Broker: Sending cloudlet 24 to VM #0
0.2: Broker: Sending cloudlet 25 to VM #1
0.2: Broker: Sending cloudlet 26 to VM #2
0.2: Broker: Sending cloudlet 27 to VM #3
0.2: Broker: Sending cloudlet 28 to VM #4
```

```
0.2: Broker: Sending cloudlet 29 to VM #5
0.2: Broker: Sending cloudlet 30 to VM #6
0.2: Broker: Sending cloudlet 31 to VM #7
0.2: Broker: Sending cloudlet 32 to VM #8
0.2: Broker: Sending cloudlet 33 to VM #9
0.2: Broker: Sending cloudlet 34 to VM #10
0.2: Broker: Sending cloudlet 35 to VM #11
0.2: Broker: Sending cloudlet 36 to VM #0
0.2: Broker: Sending cloudlet 37 to VM #1
0.2: Broker: Sending cloudlet 38 to VM #2
0.2: Broker: Sending cloudlet 39 to VM #3
```

Συνολικά, εκτελέστηκαν και τα 40 Cloudlets που ανατέθηκαν στις 12 VM που δημιουργήθηκαν. Αναλυτικά, η κατάσταση των Cloudlets έχει ως εξής:

1. **VM #0**

   Cloudlet #0, #12, #24, #36 (Συνολικά εκτελέστηκαν 4 Cloudlets σ' αυτή την VM)

2. **VM #1**

   Cloudlet #1, #13, #25, #37 (Συνολικά εκτελέστηκαν 4 Cloudlets σ' αυτή την VM)

3. **VM #2**

   Cloudlet #2, #14, #26, #38 (Συνολικά εκτελέστηκαν 4 Cloudlets σ' αυτή την VM)

4. **VM #3**

   Cloudlet #3, #15, #27, #39 (Συνολικά εκτελέστηκαν 4 Cloudlets σ' αυτή την VM)

5. **VM #4**

   Cloudlet #4, #16, #28 (Συνολικά εκτελέστηκαν 3 Cloudlets σ' αυτή την VM)

6. **VM #5**

   Cloudlet #5, #17, #29 (Συνολικά εκτελέστηκαν 4 Cloudlets σ' αυτή την VM)

7. **VM #6**

   Cloudlet #6, #18, #30 (Συνολικά εκτελέστηκαν 3 Cloudlets σ' αυτή την VM)

8. **VM #7**

   Cloudlet #7, #19, #31 (Συνολικά εκτελέστηκαν 3 Cloudlets σ' αυτή την VM)

9. **VM #8**

   Cloudlet #8, #20, #32 (Συνολικά εκτελέστηκαν 3 Cloudlets σ' αυτή την VM)

10. **VM #9**

    Cloudlet #9, #21, #33 (Συνολικά εκτελέστηκαν 3 Cloudlets σ' αυτή την VM)

11. **VM #10**

    Cloudlet #10, #22, #34 (Συνολικά εκτελέστηκαν 3 Cloudlets σ' αυτή την VM)

12. **VM #11**

    Cloudlet #11, #23, #35 (Συνολικά εκτελέστηκαν 3 Cloudlets σ' αυτή την VM)

CloudSimExample6/Output

```
3.1980000000000004: Broker: Cloudlet 4 received
3.1980000000000004: Broker: Cloudlet 16 received
3.1980000000000004: Broker: Cloudlet 28 received
3.1980000000000004: Broker: Cloudlet 5 received
3.1980000000000004: Broker: Cloudlet 17 received
3.1980000000000004: Broker: Cloudlet 29 received
3.1980000000000004: Broker: Cloudlet 6 received
3.1980000000000004: Broker: Cloudlet 18 received
3.1980000000000004: Broker: Cloudlet 30 received
3.1980000000000004: Broker: Cloudlet 7 received
3.1980000000000004: Broker: Cloudlet 19 received
3.1980000000000004: Broker: Cloudlet 31 received
3.1980000000000004: Broker: Cloudlet 8 received
3.1980000000000004: Broker: Cloudlet 20 received
3.1980000000000004: Broker: Cloudlet 32 received
3.1980000000000004: Broker: Cloudlet 10 received
3.1980000000000004: Broker: Cloudlet 22 received
3.1980000000000004: Broker: Cloudlet 34 received
3.1980000000000004: Broker: Cloudlet 9 received
3.1980000000000004: Broker: Cloudlet 21 received
3.1980000000000004: Broker: Cloudlet 33 received
3.1980000000000004: Broker: Cloudlet 11 received
3.1980000000000004: Broker: Cloudlet 23 received
3.1980000000000004: Broker: Cloudlet 35 received
4.198: Broker: Cloudlet 0 received
4.198: Broker: Cloudlet 12 received
4.198: Broker: Cloudlet 24 received
4.198: Broker: Cloudlet 36 received
4.198: Broker: Cloudlet 1 received
4.198: Broker: Cloudlet 13 received
4.198: Broker: Cloudlet 25 received
4.198: Broker: Cloudlet 37 received
4.198: Broker: Cloudlet 2 received
```

```
4.198: Broker: Cloudlet 14 received
4.198: Broker: Cloudlet 26 received
4.198: Broker: Cloudlet 38 received
4.198: Broker: Cloudlet 3 received
4.198: Broker: Cloudlet 15 received
4.198: Broker: Cloudlet 27 received
4.198: Broker: Cloudlet 39 received
4.198: Broker: All Cloudlets executed. Finishing...
```

9. Με τι πολιτική (space sharing ή time sharing – και σε ποια σημεία του κώδικα φαίνεται / προσδιορίζεται αυτό) γίνεται (α) η δρομολόγηση των Cloudlets στα VMs στα οποία τελικά ανατέθηκαν να τρέχουν; και (β) η δρομολόγηση των VMs στα PEs (ή αλλιώς, η ανάθεση των PEs στα VMs) του Host στον οποίον τρέχουν;

Στην τρέχουσα φάση της προσομοίωσης του παραδείγματος CloudSimExample6, ο αλγόριθμος που:

*(α) δρομολογεί τα Cloudlets στις εικονικές μηχανές (VMs) είναι ο χρονοπρογραμματισμός με κοινόχρηστο χρόνο (**Time-Shared Cloudlet Scheduling**).*

Στην δρομολόγηση των Cloudlets με διαμοιρασμό χρόνου, πολλά Cloudlets μπορούν να εκτελούνται παράλληλα σ' ένα VM. Αυτό είναι εφικτό με την προϋπόθεση το VM να διαθέτει επαρκείς υπολογιστικούς πόρους για την εξυπηρέτηση πολλαπλών Cloudlets παράλληλα. Η κλάση που αναπαρίστανται ο αλγόριθμος αυτός είναι η **CloudletSchedulerTimeShared.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createVM(int userId, int vms) : List<Vm>

```
//create VMs
            Vm[] vm = new Vm[vms];

            for(int i=0;i<vms;i++){
                  vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
                  //for creating a VM with a space shared scheduling policy for
cloudlets:
                  //vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, priority,
vmm, new CloudletSchedulerSpaceShared());

                  list.add(vm[i]);
            }
```

*(β) δρομολογεί τα VMs στα στοιχεία επεξεργασίας (PEs) των Hosts είναι ο χρονοπρογραμματισμός με κοινόχρηστο χρόνο (**Time-Shared VM Scheduling**).*

Στον χρονοπρογραμματισμό VM με κοινόχρηστο χρόνο, οι εικονικές μηχανές μοιράζονται τα στοιχεία επεξεργασίας (PEs) του Host που τους φιλοξενεί. Εφόσον, ένα στοιχείο επεξεργασίας

(PE) μπορεί να ικανοποιεί τις απαιτήσεις πολλών VM ταυτόχρονα, τότε αυτά με διαμοιρασμό χρόνου, θα εκτελούνται παράλληλα στο ίδιο PE του Host ή σ' ένα σύνολο από PEs του ίδιου Host. Η κλάση που αναπαρίστανται ο αλγόριθμος αυτός είναι η **VmSchedulerTimeShared.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
hostList.add(
                    new Host(
                            hostId,
                            new RamProvisionerSimple(ram),
                            new BwProvisionerSimple(bw),
                            storage,
                            peList1,
                            new VmSchedulerTimeShared(peList1)
                    )
            ); // This is our first machine

            hostId++;

            hostList.add(
                    new Host(
                            hostId,
                            new RamProvisionerSimple(ram),
                            new BwProvisionerSimple(bw),
                            storage,
                            peList2,
                            new VmSchedulerTimeShared(peList2)
                    )
            ); // Second machine
```

10. Εξηγείστε αναλυτικά το τελικό μέρος των αποτελεσμάτων (πίνακα output στο τέλος) του παραδείγματος - τι δίνει η κάθε στήλη κλπ, και ειδικότερα μεταξύ των άλλων τους χρόνους εκκίνησης και περάτωσης των Cloudlets (σε άμεση συσχέτιση με τις πολιτικές δρομολόγησης που παρατηρήσατε ότι ακολουθούνται στο ερώτημα 9).

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες:

1. **Cloudlet ID**

   Πρόκειται για το Cloudlet που ανατέθηκε στο αντίστοιχο VM για εκτέλεση. Η αναγνώριση του καθορίζεται από το entity_id που του δίνεται κατά την δημιουργία του. Τα id κυμαίνονται από #0 εώς #39, καθώς, δημιουργήθηκαν συνολικά 40 Cloudlets (βλ. ερώτημα 7).

2. **Status**

   Πρόκειται για την κατάσταση εκτέλεσης των Cloudlets. Με τον όρο SUCCESS σημαίνει ότι το Cloudlet εκτελέστηκε με επιτυχία, ενώ με τον όρο FAIL σημαίνει το αντίθετο.

3. **Data center ID**

   Πρόκειται για το Datacenter μέσα στο οποίο βρίσκεται το VM που εκτέλεσε το αντίστοιχο Cloudlet. Η αναγνώριση του καθορίζεται από το entity_id που του δίνεται κατά την δημιουργία του. Τα id κυμαίνονται από #2 εώς #3, καθώς, δημιουργήθηκαν συνολικά 2 Datacenters (βλ. ερώτημα 1).

4. **VM ID**

   Πρόκειται για το VM μέσα στο οποίο εκτελείται το αντίστοιχο Cloudlet. Η αναγνώριση του καθορίζεται από το entity_id που του δίνεται κατά την δημιουργία του. Τα id κυμαίνονται από #0 εώς #19, καθώς, δημιουργήθηκαν συνολικά 20 VMs (βλ. ερώτημα 4).

5. **Time**

   Πρόκειται για τον χρόνο που δαπανήθηκε από τον επεξεργαστή του VM για την εκτέλεση του αντίστοιχου Cloudlet. Ο χρόνος αυτός μετριέται σε **seconds** και ο υπολογισμός του βγαίνει από την εξίσωση

$$\text{Time} = \text{Finish Time} - \text{Start Time}$$

6. **Start Time**

   Πρόκειται για τον χρόνο εκκίνησης της εκτέλεσης του αντίστοιχου Cloudlet (μετριέται σε seconds).

7. **Finish Time**

   Πρόκειται για τον χρόνο περάτωσης της εκτέλεσης του αντίστοιχου Cloudlet (μετριέται σε seconds).

CloudSimExample6/Output

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    4        SUCCESS        2            4            3        0,2            3,2
    16        SUCCESS        2            4            3        0,2            3,2
    28        SUCCESS        2            4            3        0,2            3,2
    5        SUCCESS        2            5            3        0,2            3,2
    17        SUCCESS        2            5            3        0,2            3,2
    29        SUCCESS        2            5            3        0,2            3,2
    6        SUCCESS        3            6            3        0,2            3,2
    18        SUCCESS        3            6            3        0,2            3,2
    30        SUCCESS        3            6            3        0,2            3,2
    7        SUCCESS        3            7            3        0,2            3,2
    19        SUCCESS        3            7            3        0,2            3,2
    31        SUCCESS        3            7            3        0,2            3,2
    8        SUCCESS        3            8            3        0,2            3,2
    20        SUCCESS        3            8            3        0,2            3,2
    32        SUCCESS        3            8            3        0,2            3,2
    10        SUCCESS        3            10            3        0,2            3,2
    22        SUCCESS        3            10            3        0,2            3,2
    34        SUCCESS        3            10            3        0,2            3,2
    9        SUCCESS        3            9            3        0,2            3,2
    21        SUCCESS        3            9            3        0,2            3,2
    33        SUCCESS        3            9            3        0,2            3,2
    11        SUCCESS        3            11            3        0,2            3,2
    23        SUCCESS        3            11            3        0,2            3,2
    35        SUCCESS        3            11            3        0,2            3,2
    0        SUCCESS        2            0            4        0,2            4,2
    12        SUCCESS        2            0            4        0,2            4,2
    24        SUCCESS        2            0            4        0,2            4,2
    36        SUCCESS        2            0            4        0,2            4,2
    1        SUCCESS        2            1            4        0,2            4,2
    13        SUCCESS        2            1            4        0,2            4,2
    25        SUCCESS        2            1            4        0,2            4,2
    37        SUCCESS        2            1            4        0,2            4,2
    2        SUCCESS        2            2            4        0,2            4,2
    14        SUCCESS        2            2            4        0,2            4,2
    26        SUCCESS        2            2            4        0,2            4,2
    38        SUCCESS        2            2            4        0,2            4,2
    3        SUCCESS        2            3            4        0,2            4,2
    15        SUCCESS        2            3            4        0,2            4,2
    27        SUCCESS        2            3            4        0,2            4,2
    39        SUCCESS        2            3            4        0,2            4,2
```

```
CloudSimExample6 finished!
```

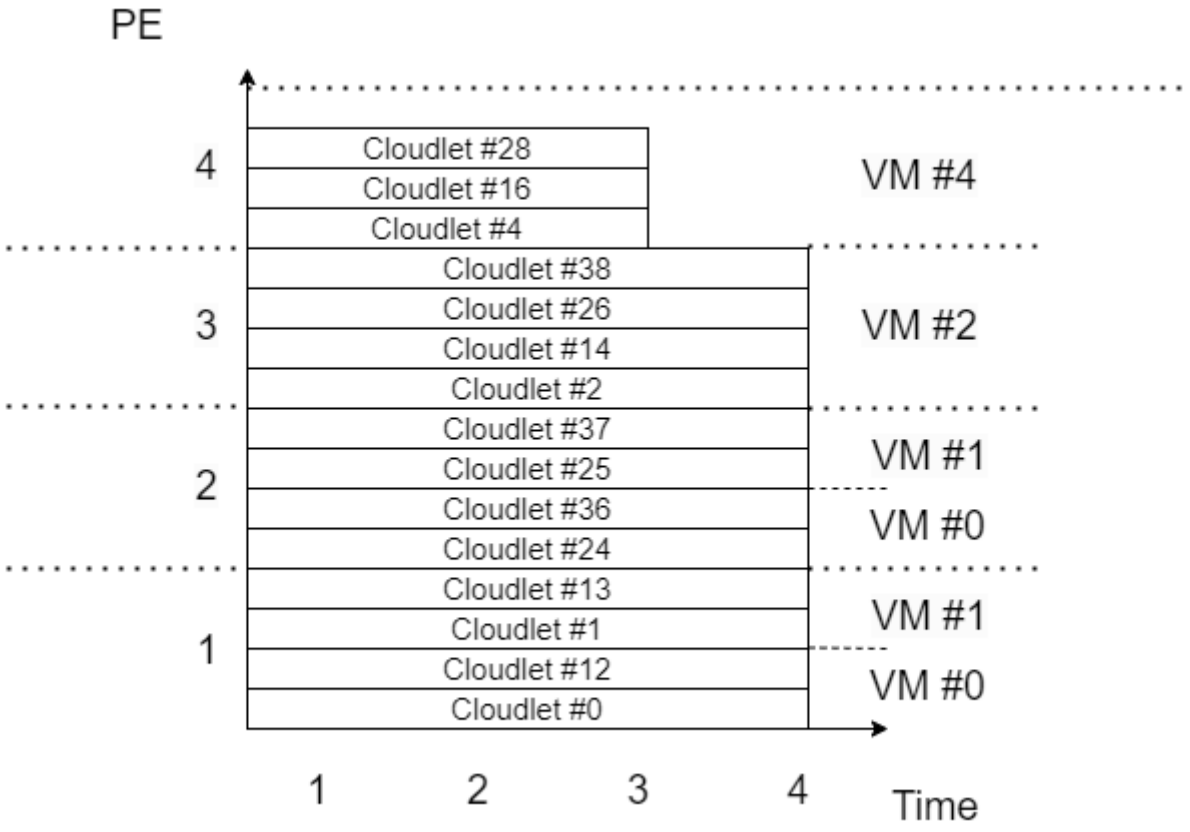
Οι χρόνοι που απεικονίζονται στην στήλη **Time** κυμαίνονται στα 3 και 4 seconds. Αν αναλογιστούμε ότι η πολιτική δρομολόγησης των Cloudlets στα VMs και των VMs στα PEs των Hosts είναι η **Time-Shared** πολιτική (αναλυτικά βλ. ερώτημα 9), οι χρόνοι είναι αναμενόμενα μικροί. Αυτό δεν οφείλεται μόνο στην πολιτική δρομολόγησης, αλλά και στο πλήθος των Cloudlets που έχει το κάθε VM, καθώς και στα αντίστοιχα χαρακτηριστικά του Host που φιλοξενεί το VM, τα χαρακτηριστικά του VM και του Cloudlet που εκτελείται.

Όπως απαντήθηκε στο ερώτημα 8, τα Cloudlets που έχουν ανατεθεί στα VMs #0, #1, #2, #3 χρειάστηκαν 4 seconds εκάστως για την εκτέλεση τους, ενώ στα υπόλοιπα VMs (#4 ... #19) χρειάστηκαν 3 seconds εκάστως. Η απάντηση στην μικρή διαφορά χρόνων έχει να κάνει στην δημιουργία των 40 Cloudlets και στην κατανομή τους στα 12 VMs που δημιουργήθηκαν με επιτυχία στους αντίστοιχους Hosts. Η πολιτική ανάθεσης των Cloudlets στα VMs είναι η **First-Fit**, συνεπώς, οι πρώτες 4 VM θα έχουν ένα Cloudlet παραπάνω από τα υπόλοιπα 16 VMs.

Ας πάρουμε για παράδειγμα τον Host #0 του Datacenter #2.

Ο Host #0 είναι μία φυσική 4-πύρηνη μηχανή με απόδοση 1000 MIPS ανά πυρήνα. Φιλοξενεί συνολικά 4 VMs που είναι 1-πύρηνοι με απόδοση 1000 MIPS εκάστως. Με βάση την πολιτική **First-Fit**, στα VM #0, #1, #2 έχουν ανατεθεί 4 Cloudlets μεγέθους 1000 MI το καθένα, ενώ στο VM #4 έχουν ανατεθεί 3 Cloudlets μεγέθους 1000 MI το καθένα.

Με την πολιτική δρομολόγησης **Time-Shared Cloudlet Scheduling** και **Time-Shared VM Scheduling**, τα Cloudlets εκτελούνται ταυτόχρονα στα VMs, και τα VMs χρησιμοποιούν παράλληλα τον ίδιο πυρήνα του Host, όπως φαίνεται και στην **Εικόνα 10.1.**



Εικόνα 10.1 Time-Shared Cloudlet Scheduling και Time-Shared VM Scheduling στον Host #0 του Datacenter #2

Ο κάθε πυρήνας του Host χρησιμοποιείται ταυτόχρονα από 1-2 VMs που εκτελούν 2-3 Cloudlets το κάθενα. Με το μέγεθος του κάθε Cloudlet να είναι στα 1000 MI και την απόδοση του κάθε πυρήνα του Host να είναι 1000 MIPS, το κάθε Cloudlet εκτελείται με ρυθμό

$$\text{Rate} = \text{MIPS} / \#\text{Cloudlets}$$

, όπου MIPS η απόδοση του πυρήνα του Host που εκτελείται το Cloudlet, και #Cloudlets ο αριθμός των Cloudlets που εκτελούνται στον πυρήνα του Host. Αναλυτικά έχουμε:

- **Πυρήνας 1**

  $$\text{Rate} = \text{MIPS} / \#\text{Cloudlets} \rightarrow \text{Rate} = 1000 / 4 \rightarrow \text{Rate} = 250 \text{ MIPS}$$

  Το κάθε Cloudlet των 1000 MI εκτελείται με ρυθμό 250 MIPS, συνεπώς, θα χρειαστεί 4 second για την εκτέλεση τους.

- **Πυρήνας 2**

  Παρόμοια, όπως ισχύει και στον Πυρήνα 1

- **Πυρήνας 3**

  Παρόμοια, όπως ισχύει και στον Πυρήνα 1

- **Πυρήνας 4**

  $$\text{Rate} = \text{MIPS} / \#\text{Cloudlets} \rightarrow \text{Rate} = 1000 / 3 \rightarrow \text{Rate} \cong 333 \text{ MIPS}$$

  Το κάθε Cloudlet των 1000 MI εκτελείται περίπου με ρυθμό περίπου 333 MIPS, συνεπώς, θα χρειαστεί 3 second για την εκτέλεση τους.

Τέλος, παρατηρούμε ότι οι χρόνοι εκκίνησης των Cloudlets είναι από 0.2 second αντί για 0.0 second, καθώς δαπανάται**,** ένας χρόνος για την δημιουργία των VMs και την ανάθεση τους στους Hosts των Datacenters, καθώς την ανάθεση των Cloudlets στα VMs. Η οντότητα που διαχείριζεται αυτές τις διεργασίες ονομάζεται **Broker** και αναπαρίστανται στην κλάση **DatacenterBroker.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

Όπως βλέπουμε και στο output της προσομοίωσης, 0.1 seconds χρειάστηκαν για την δημιουργία των VMs και την ανάθεση τους στους Hosts του Datacenter #2.

CloudSimExample6/Output

```
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
```

Όπως επίσης, βλέπουμε και στο output της προσομοίωσης, άλλα 0.1 seconds χρειάστηκαν για την δημιουργία των VMs και την ανάθεση τους στους Hosts του Datacenter #3, καθώς, και στη δημιουργία των Cloudlets και της ανάθεσης τους στα VMs.

CloudSimExample6/Output

```
0.2: Broker: VM #6 has been created in Datacenter #3, Host #0
0.2: Broker: VM #7 has been created in Datacenter #3, Host #0
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #1
0.2: Broker: VM #10 has been created in Datacenter #3, Host #0
0.2: Broker: VM #11 has been created in Datacenter #3, Host #1
...
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
0.2: Broker: Sending cloudlet 2 to VM #2
0.2: Broker: Sending cloudlet 3 to VM #3
0.2: Broker: Sending cloudlet 4 to VM #4
```

Πρακτικό Μέρος

Α. Τρέξτε ξανά το παράδειγμα με τις υπόλοιπες εναλλακτικές δυνατότητες που σας παρέχονται (space/time sharing) για τα σημεία 9(α) και 9(β) παραπάνω, δηλαδή (α) για τη δρομολόγηση των Cloudlets στα VMs, και (β) για τη δρομολόγηση των VMs στους Hosts. Μελετήστε πάλι το output σε κάθε περίπτωση, δείτε αν υπάρχουν ή όχι διαφορές, και εξηγείστε το γιατί. Τρέξτε επίσης ξανά το παράδειγμα εφαρμόζοντας για τη δρομολόγηση των VMs πολιτική time sharing με over-subscription και περιγράψτε-εξηγείστε πάλι το output.

Όπως απαντήθηκε στο ερώτημα 9, οι πολιτικές δρομολόγησης που ακολουθεί η προσομοίωση του παραδείγματος **CloudSimExample6** είναι η

(α) Time-Shared Cloudlet Scheduling

(β) Time-Shared VM Scheduling

Ας εξετάσουμε και τις υπόλοιπες εναλλακτικές δυνατότητες που μας παρέχονται:

(α) Space-Shared Cloudlet Scheduling & Time-Shared VM Scheduling (SC-TV)

Στον χρονοπρογραμματισμό Cloudlet με διαμοιρασμό χώρου (**Space-Shared Cloudlet Scheduling**), εάν ένα Cloudlet ανατεθεί σ' ένα VM για εκτέλεση, τότε κανένα άλλο Cloudlet δεν θα εκτελεστεί από αυτό το VM μέχρι το τρέχον Cloudlet να αποδεσμευτεί από το VM. Πολλαπλά Cloudlets δεν μπορούν να εκτελεστούν παράλληλα σε ένα VM, ακόμη και αν το VM είναι αρκετά ικανό για κάτι τέτοιο. Η κλάση που αναπαρίστανται ο αλγόριθμος αυτός είναι η **CloudletSchedulerSpaceShared.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createVM(int userId, int vms) : List<Vm>

```
//create VMs
            Vm[] vm = new Vm[vms];

            for(int i=0;i<vms;i++){
                  //vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
                  //for creating a VM with a space shared scheduling policy for
cloudlets:
                  vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerSpaceShared());

                  list.add(vm[i]);
```

```
        }
```

Ο χρονοπρογραμματισμός VM με διαμοιρασμό χρόνου (**Time-Shared VM Scheduling**), αναφέρεται με σαφήνεια στο ερώτημα 9.

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες (για επεξήγηση των στηλών προσομοίωσης, βλ. ερώτημα 10)

CloudSimExample6/Output

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish
Time
    0        SUCCESS        2            0            1        0,2            1,2
    1        SUCCESS        2            1            1        0,2            1,2
    2        SUCCESS        2            2            1        0,2            1,2
    4        SUCCESS        2            4            1        0,2            1,2
    3        SUCCESS        2            3            1        0,2            1,2
    5        SUCCESS        2            5            1        0,2            1,2
    6        SUCCESS        3            6            1        0,2            1,2
    7        SUCCESS        3            7            1        0,2            1,2
    8        SUCCESS        3            8            1        0,2            1,2
    10        SUCCESS        3            10            1        0,2            1,2
    9        SUCCESS        3            9            1        0,2            1,2
    11        SUCCESS        3            11            1        0,2            1,2
    12        SUCCESS        2            0            1        1,2            2,2
    13        SUCCESS        2            1            1        1,2            2,2
    14        SUCCESS        2            2            1        1,2            2,2
    16        SUCCESS        2            4            1        1,2            2,2
    15        SUCCESS        2            3            1        1,2            2,2
    17        SUCCESS        2            5            1        1,2            2,2
    18        SUCCESS        3            6            1        1,2            2,2
    19        SUCCESS        3            7            1        1,2            2,2
    20        SUCCESS        3            8            1        1,2            2,2
    22        SUCCESS        3            10            1        1,2            2,2
    21        SUCCESS        3            9            1        1,2            2,2
    23        SUCCESS        3            11            1        1,2            2,2
    24        SUCCESS        2            0            1        2,2            3,2
    25        SUCCESS        2            1            1        2,2            3,2
    26        SUCCESS        2            2            1        2,2            3,2
    28        SUCCESS        2            4            1        2,2            3,2
    27        SUCCESS        2            3            1        2,2            3,2
    29        SUCCESS        2            5            1        2,2            3,2
    30        SUCCESS        3            6            1        2,2            3,2
    31        SUCCESS        3            7            1        2,2            3,2
    32        SUCCESS        3            8            1        2,2            3,2
    34        SUCCESS        3            10            1        2,2            3,2
    33        SUCCESS        3            9            1        2,2            3,2
    35        SUCCESS        3            11            1        2,2            3,2
    36        SUCCESS        2            0            1        3,2            4,2
    37        SUCCESS        2            1            1        3,2            4,2
    38        SUCCESS        2            2            1        3,2            4,2
    39        SUCCESS        2            3            1        3,2            4,2
CloudSimExample6 finished!
```

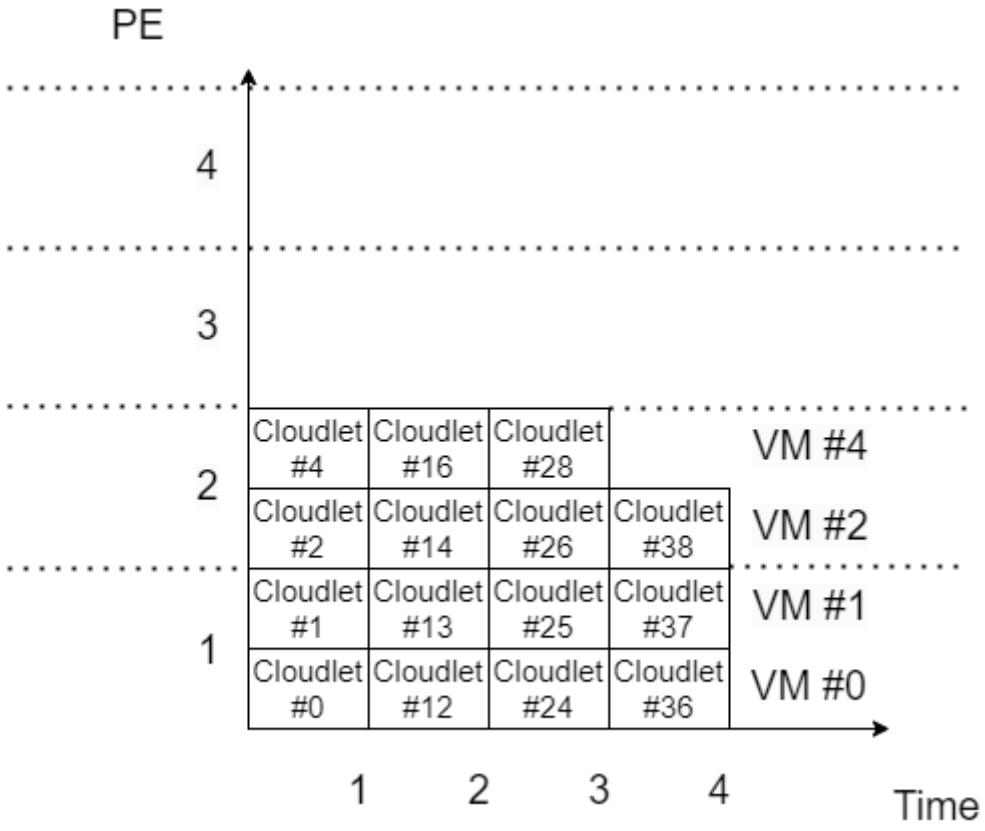
Η διαφορά στους χρόνους εκτέλεσης των Cloudlets της πολιτικής **SC-TV** με τους αντίστοιχους χρόνους της πολιτικής **TC-TV** που είδαμε στο ερώτημα 10, είναι φανερή, διότι, το κάθε Cloudlet δεσμεύεται αποκλειστικά το στοιχείο επεξεργασίας του VM, χωρίς να επιτρέπεται να δεσμευτεί από άλλο μέχρι την ολοκλήρωση της εκτέλεσης του. Επομένως, το κάθε Cloudlet μεγέθους 1000 MI θα δεσμευτεί αποκλειστικά τον πυρήνα του VM απόδοσης 1000 MIPS, γεγονός που αιτιολογεί τον χρόνο εκτέλεσης του κάθε Cloudlet στα 1 second.

Παρατηρούμε επίσης, και διαφορά ως προς τους χρόνους εκκίνησης και ολοκλήρωσης του κάθε Cloudlet, καθώς, δεν έχει οριστεί κάποια προτεραιότητα και γι' αυτό τον λόγο τα Cloudlets δεσμεύονται τους πυρήνες των VM τους με την σειρά κατά αύξουσα σειρά entity_id. Συνεπώς, ο χρόνος εκκίνησης του επόμενου Cloudlet εξαρτάται αποκλειστικά από τον χρόνο ολοκλήρωσης του αμέσως προηγούμενου Cloudlet που δεσμεύτηκε τον πυρήνα του VM.

Ας πάρουμε για παράδειγμα τον Host #0 του Datacenter #2.

Ο Host #0 είναι μία φυσική 4-πύρηνη μηχανή με απόδοση 1000 MIPS ανά πυρήνα. Φιλοξενεί συνολικά 4 VMs που είναι 1-πύρηνοι με απόδοση 1000 MIPS εκάστως. Με βάση την πολιτική **First-Fit**, στα VM #0, #1, #2 έχουν ανατεθεί 4 Cloudlets μεγέθους 1000 MI το καθένα, ενώ στο VM #4 έχουν ανατεθεί 3 Cloudlets μεγέθους 1000 MI το καθένα.

Με την πολιτική δρομολόγησης **Space-Shared Cloudlet Scheduling (SC)** και **Time-Shared VM Scheduling (TV)**, τα Cloudlets δεν εκτελούνται ταυτόχρονα στα VMs σε αντίθεση με τα VMs που χρησιμοποιούν παράλληλα τον ίδιο πυρήνα του Host, όπως φαίνεται και στην **Εικόνα Α.1**



Εικόνα Α.1 Space-Shared Cloudlet Scheduling και Time-Shared VM Scheduling στον Host #0 του Datacenter #2

(β) Space-Shared Cloudlet Scheduling & Space-Shared VM Scheduling (SC-SV)

Ο χρονοπρογραμματισμός Cloudlet με διαμοιρασμό χώρου (**Space-Shared Cloudlet Scheduling**), αναφέρεται με σαφήνεια στη περίπτωση α.

Στον χρονοπρογραμματισμό VM με διαμοιρασμό χώρου (**Space-Shared VM Scheduling**), τα στοιχεία επεξεργασίας (PE) κατανέμονται σ' ένα μόνο VM. Διεξοδικά, σημαίνει ότι αν ένα PE καταχωρηθεί σ' ένα VM, δεν είναι δυνατή η πρόσβαση σ' αυτό από άλλο VM, μέχρι αυτό να αποδεσμευτεί. Η κλάση που αναπαρίστανται ο αλγόριθμος αυτός είναι η **VmSchedulerSpaceShared.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
//To create a host with a space-shared allocation policy for PEs to VMs:
            hostList.add(
                  new Host(
                        hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                        storage,
                        peList1,
                        new VmSchedulerSpaceShared(peList1)
                  )
            );

            hostId++;

            //To create a host with a space-shared allocation policy for PEs to VMs:
                        hostList.add(
                              new Host(
                                    hostId,
                                    new RamProvisionerSimple(ram),
                                    new BwProvisionerSimple(bw),
                                    storage,
                                    peList2,
                                    new VmSchedulerSpaceShared(peList2)
                              )
                        );
```

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες (βλ. ερώτημα 10)

CloudSimExample6/Output

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish
Time
    0        SUCCESS        2            0            1        0,2            1,2
    1        SUCCESS        2            1            1        0,2            1,2
    2        SUCCESS        2            2            1        0,2            1,2
```

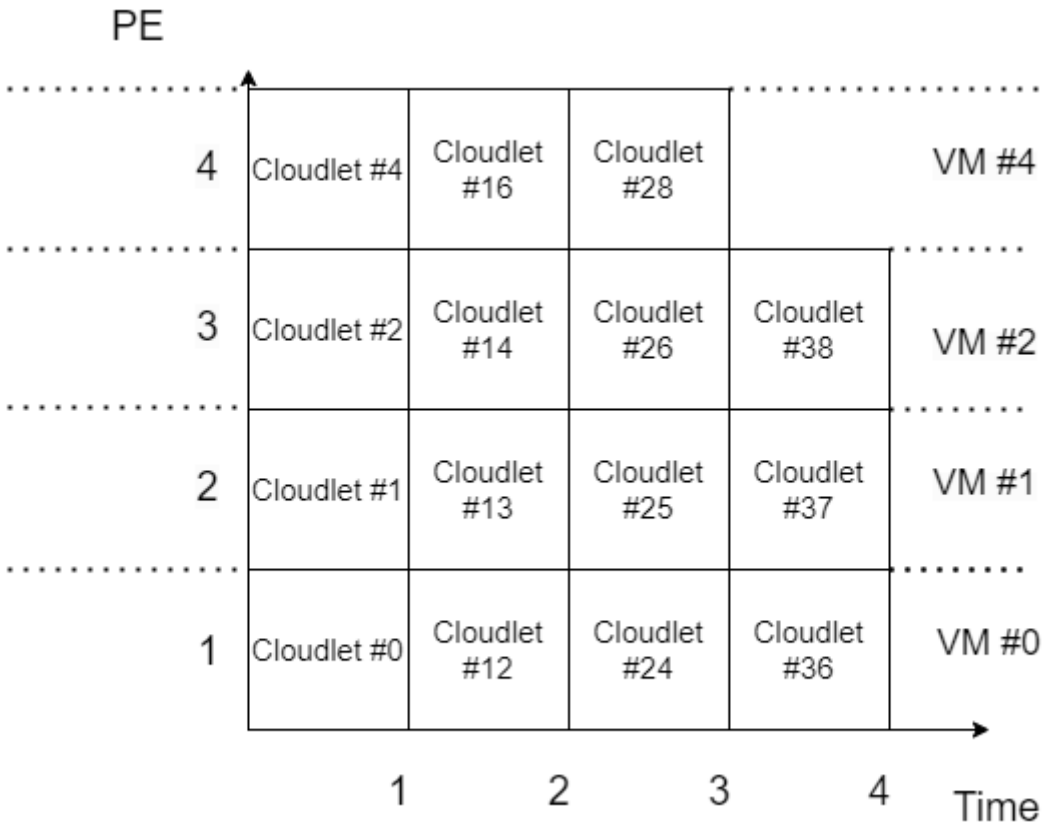
```
    4         SUCCESS         2          4          1         0,2          1,2
    3         SUCCESS         2          3          1         0,2          1,2
    5         SUCCESS         2          5          1         0,2          1,2
    6         SUCCESS         3          6          1         0,2          1,2
    7         SUCCESS         3          7          1         0,2          1,2
    8         SUCCESS         3          8          1         0,2          1,2
    10         SUCCESS         3          10          1         0,2          1,2
    9         SUCCESS         3          9          1         0,2          1,2
    11         SUCCESS         3          11          1         0,2          1,2
    12         SUCCESS         2          0          1         1,2          2,2
    13         SUCCESS         2          1          1         1,2          2,2
    14         SUCCESS         2          2          1         1,2          2,2
    16         SUCCESS         2          4          1         1,2          2,2
    15         SUCCESS         2          3          1         1,2          2,2
    17         SUCCESS         2          5          1         1,2          2,2
    18         SUCCESS         3          6          1         1,2          2,2
    19         SUCCESS         3          7          1         1,2          2,2
    20         SUCCESS         3          8          1         1,2          2,2
    22         SUCCESS         3          10          1         1,2          2,2
    21         SUCCESS         3          9          1         1,2          2,2
    23         SUCCESS         3          11          1         1,2          2,2
    24         SUCCESS         2          0          1         2,2          3,2
    25         SUCCESS         2          1          1         2,2          3,2
    26         SUCCESS         2          2          1         2,2          3,2
    28         SUCCESS         2          4          1         2,2          3,2
    27         SUCCESS         2          3          1         2,2          3,2
    29         SUCCESS         2          5          1         2,2          3,2
    30         SUCCESS         3          6          1         2,2          3,2
    31         SUCCESS         3          7          1         2,2          3,2
    32         SUCCESS         3          8          1         2,2          3,2
    34         SUCCESS         3          10          1         2,2          3,2
    33         SUCCESS         3          9          1         2,2          3,2
    35         SUCCESS         3          11          1         2,2          3,2
    36         SUCCESS         2          0          1         3,2          4,2
    37         SUCCESS         2          1          1         3,2          4,2
    38         SUCCESS         2          2          1         3,2          4,2
    39         SUCCESS         2          3          1         3,2          4,2
CloudSimExample6 finished!
```

Οι χρόνοι είναι ακριβώς ίδιοι με την περίπτωση 2 της πολιτικής SC-TV. Αυτό συμβαίνει λόγω της πολιτικής **Space-Shared VM Scheduling**, όπου ένα VM καταλαμβάνει ένα επεξεργαστικό στοιχείο μέχρι να ολοκληρώσει όλα τα Cloudlets του. Αν δεν υπάρχει άλλο ελεύθερο στοιχείο επεξεργασίας, τα άλλα VMs πρέπει να περιμένουν μέχρι να τελειώσουν όλα τα Cloudlets του τρέχοντος VM. Η πολιτική Space-Shared Cloudlet Scheduling αυξάνει ακόμα περισσότερο τον χρόνο αναμονής για το επόμενο VM που περιμένει να χρησιμοποιήσει το επεξεργαστικό στοιχείο του Host.

Ας πάρουμε για παράδειγμα τον Host #0 του Datacenter #2.

Ο Host #0 είναι μία φυσική 4-πύρηνη μηχανή με απόδοση 1000 MIPS ανά πυρήνα. Φιλοξενεί συνολικά 4 VMs που είναι 1-πύρηνοι με απόδοση 1000 MIPS εκάστως. Με βάση την πολιτική **First-Fit**, στα VM #0, #1, #2 έχουν ανατεθεί 4 Cloudlets μεγέθους 1000 MI το καθένα, ενώ στο VM #4 έχουν ανατεθεί 3 Cloudlets μεγέθους 1000 MI το καθένα.

Με την πολιτική δρομολόγησης **Space-Shared Cloudlet Scheduling (SC)** και **Space-Shared VM Scheduling (SV)**, τα Cloudlets δεν εκτελούνται ταυτόχρονα στα VMs, ούτε τα VMs χρησιμοποιούν παράλληλα τον ίδιο πυρήνα του Host, όπως φαίνεται και στην **Εικόνα Α.2**



Εικόνα Α.2 Space-Shared Cloudlet Scheduling και Space-Shared VM Scheduling στον Host #0 του Datacenter #2

(γ) Time-Shared Cloudlet Scheduling & Space-Shared VM Scheduling (TC-SV)

Ο χρονοπρογραμματισμός Cloudlet με διαμοιρασμό χρόνου (**Time-Shared Cloudlet Scheduling**), αναφέρεται με σαφήνεια στο ερώτημα 9.

Ο χρονοπρογραμματισμός VM με διαμοιρασμό χώρου (**Space-Shared VM Scheduling**), αναφέρεται με σαφήνεια στη περίπτωση β.

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες (για επεξήγηση των στηλών προσομοίωσης, βλ. ερώτημα 10)

CloudSimExample6/Output
========== OUTPUT ==========

```
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish
Time
    4        SUCCESS        2            4            3        0,2            3,2
    16        SUCCESS        2            4            3        0,2            3,2
    28        SUCCESS        2            4            3        0,2            3,2
    5        SUCCESS        2            5            3        0,2            3,2
    17        SUCCESS        2            5            3        0,2            3,2
    29        SUCCESS        2            5            3        0,2            3,2
    6        SUCCESS        3            6            3        0,2            3,2
    18        SUCCESS        3            6            3        0,2            3,2
    30        SUCCESS        3            6            3        0,2            3,2
    7        SUCCESS        3            7            3        0,2            3,2
    19        SUCCESS        3            7            3        0,2            3,2
    31        SUCCESS        3            7            3        0,2            3,2
    8        SUCCESS        3            8            3        0,2            3,2
    20        SUCCESS        3            8            3        0,2            3,2
    32        SUCCESS        3            8            3        0,2            3,2
    10        SUCCESS        3            10            3        0,2            3,2
    22        SUCCESS        3            10            3        0,2            3,2
    34        SUCCESS        3            10            3        0,2            3,2
    9        SUCCESS        3            9            3        0,2            3,2
    21        SUCCESS        3            9            3        0,2            3,2
    33        SUCCESS        3            9            3        0,2            3,2
    11        SUCCESS        3            11            3        0,2            3,2
    23        SUCCESS        3            11            3        0,2            3,2
    35        SUCCESS        3            11            3        0,2            3,2
    0        SUCCESS        2            0            4        0,2            4,2
    12        SUCCESS        2            0            4        0,2            4,2
    24        SUCCESS        2            0            4        0,2            4,2
    36        SUCCESS        2            0            4        0,2            4,2
    1        SUCCESS        2            1            4        0,2            4,2
    13        SUCCESS        2            1            4        0,2            4,2
    25        SUCCESS        2            1            4        0,2            4,2
    37        SUCCESS        2            1            4        0,2            4,2
    2        SUCCESS        2            2            4        0,2            4,2
    14        SUCCESS        2            2            4        0,2            4,2
    26        SUCCESS        2            2            4        0,2            4,2
    38        SUCCESS        2            2            4        0,2            4,2
    3        SUCCESS        2            3            4        0,2            4,2
    15        SUCCESS        2            3            4        0,2            4,2
    27        SUCCESS        2            3            4        0,2            4,2
    39        SUCCESS        2            3            4        0,2            4,2
CloudSimExample6 finished!
```

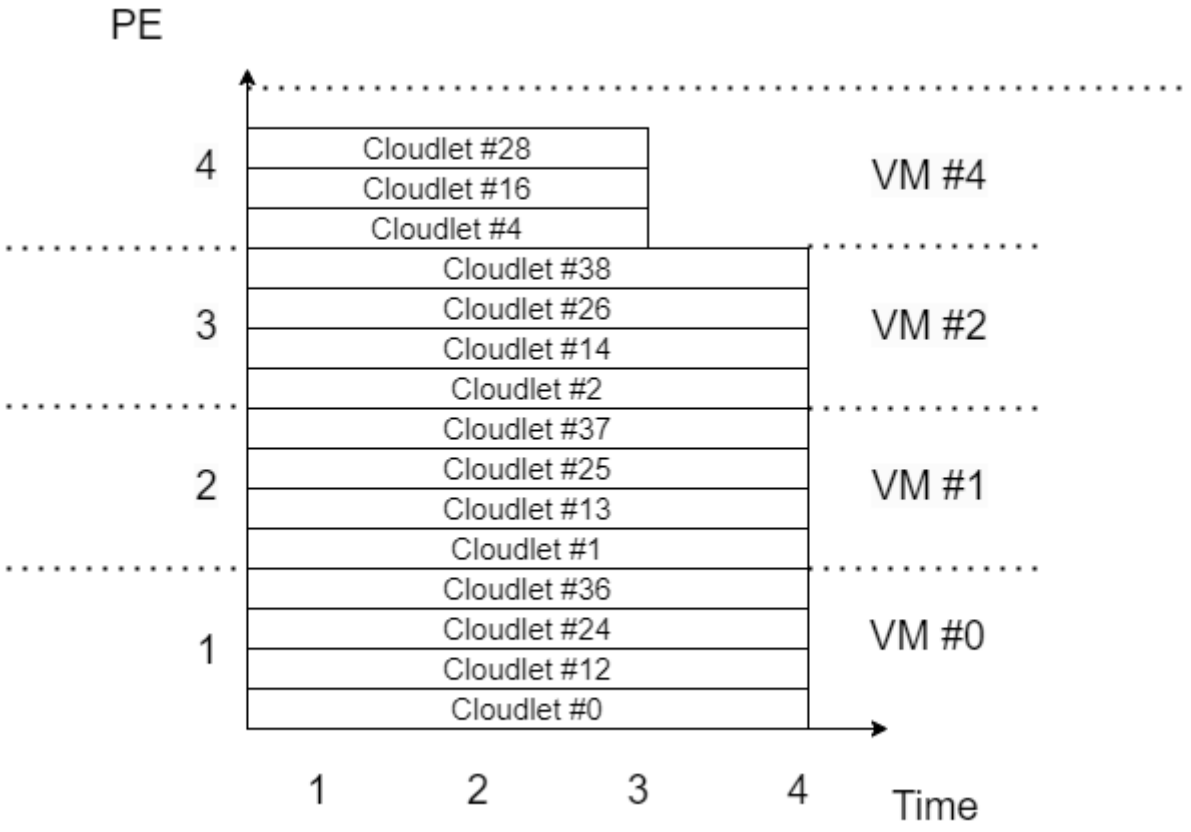
Παρατηρούμε ότι οι χρόνοι είναι ακριβώς ίδιοι με το ερώτημα 9 της πολιτικής TC-TV. Αυτό οφείλεται στην πολιτική **Space-Shared VM Scheduling**, όπου ένα VM χρησιμοποιεί ένα στοιχείο επεξεργασίας μέχρι να ολοκληρώσει όλα τα Cloudlets του. Αν δεν υπάρχει διαθέσιμο άλλο επεξεργαστικό στοιχείο, τα υπόλοιπα VMs πρέπει να περιμένουν μέχρι να τελειώσει το τρέχον VM. Επιπλέον, η πολιτική **Time-Shared Cloudlet Scheduling** κάνει την αναμονή ακόμα μεγαλύτερη για το επόμενο VM που περιμένει να χρησιμοποιήσει το επεξεργαστικό στοιχείο του Host.

Ας πάρουμε για παράδειγμα τον Host #0 του Datacenter #2.

Ο Host #0 είναι μία φυσική 4-πύρηνη μηχανή με απόδοση 1000 MIPS ανά πυρήνα. Φιλοξενεί συνολικά 4 VMs που είναι 1-πύρηνοι με απόδοση 1000 MIPS εκάστως. Με βάση την

πολιτική **First-Fit**, στα VM #0, #1, #2 έχουν ανατεθεί 4 Cloudlets μεγέθους 1000 MI το καθένα, ενώ στο VM #4 έχουν ανατεθεί 3 Cloudlets μεγέθους 1000 MI το καθένα.

Με την πολιτική δρομολόγησης **Time-Shared Cloudlet Scheduling (SC)** και **Space-Shared VM Scheduling (SV)**, τα Cloudlets κτελούνται ταυτόχρονα στα VMs, αλλά τα VMs δεν χρησιμοποιούν παράλληλα τον ίδιο πυρήνα του Host, όπως φαίνεται και στην **Εικόνα Α.3**



Εικόνα Α.3 Time-Shared Cloudlet Scheduling και Space-Shared VM Scheduling στον Host #0 του Datacenter #2

(δ) Time-Shared Cloudlet Scheduling & Time-Shared VM Scheduling με over-subscription

Ο χρονοπρογραμματισμός Cloudlet με διαμοιρασμό χρόνου (**Time-Shared Cloudlet Scheduling**), αναφέρεται με σαφήνεια στο ερώτημα 9.

Ο χρονοπρογραμματισμός VM με διαμοιρασμό χρόνου (**Time-Shared VM Scheduling**), αναφέρεται με σαφήνεια στο ερώτημα 9. Με την ιδιότητα **over-subscription** στον χρονοδρομολογητή, επιτρέπεται η κατανομή VM στον Host ακόμα και όταν απαιτεί περισσότερη χωρητικότητα επεξεργασίας από τη διαθέσιμη. Η κλάση που αναπαρίστανται ο αλγόριθμος αυτός είναι η **VmSchedulerTimeSharedOverSubscription.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
hostList.add(
                    new Host(
                          hostId,
                          new RamProvisionerSimple(ram),
                          new BwProvisionerSimple(bw),
                          storage,
                          peList1,
                          new VmSchedulerTimeSharedOverSubscription(peList1)
                    )
              ); // This is our first machine

              hostId++;

              hostList.add(
                    new Host(
                          hostId,
                          new RamProvisionerSimple(ram),
                          new BwProvisionerSimple(bw),
                          storage,
                          peList2,
                          new VmSchedulerTimeSharedOverSubscription(peList2)
                    )
              ); // Second machine
```

Η νέα κατάσταση ανάθεσης των 16 VMs που δεσμεύτηκαν με επιτυχία στους Hosts των Datacenters είναι η εξής:

2. **Datacenter_0 (#2)**

   - **Host #0**

     VM #0, #1, #2, #4 (Συνολικά 4 VMs τρέχουν σ' αυτό τον Host).

   - **Host #1**

     VM #3, #5, #6, #7 (Συνολικά 4 VMs τρέχουν σ' αυτό τον Host).

CloudSimExample6/Output

```
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
0.1: Broker: VM #6 has been created in Datacenter #2, Host #1
0.1: Broker: VM #7 has been created in Datacenter #2, Host #1
```

2. **Datacenter_1 (#3)**

- **Host #0**

  VM #8, #9, #10, #12 (Συνολικά 4 VMs τρέχουν σ' αυτό τον Host).

- **Host #1**

  VM #11, #13, #14, #15 (Συνολικά 4 VMs τρέχουν σ' αυτό τον Host)

CloudSimExample6/Output

```
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #0
0.2: Broker: VM #10 has been created in Datacenter #3, Host #0
0.2: Broker: VM #11 has been created in Datacenter #3, Host #1
0.2: Broker: VM #12 has been created in Datacenter #3, Host #0
0.2: Broker: VM #13 has been created in Datacenter #3, Host #1
0.2: Broker: VM #14 has been created in Datacenter #3, Host #1
0.2: Broker: VM #15 has been created in Datacenter #3, Host #1
```

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες (βλ. ερώτημα 10)

CloudSimExample6/Output

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish
Time
    8        SUCCESS        3            8            2        0,2            2,2
    24        SUCCESS        3            8            2        0,2            2,2
    9        SUCCESS        3            9            2        0,2            2,2
    25        SUCCESS        3            9            2        0,2            2,2
    10        SUCCESS        3            10            2        0,2            2,2
    26        SUCCESS        3            10            2        0,2            2,2
    12        SUCCESS        3            12            2        0,2            2,2
    28        SUCCESS        3            12            2        0,2            2,2
    0        SUCCESS        2            0            3        0,2            3,2
    16        SUCCESS        2            0            3        0,2            3,2
    32        SUCCESS        2            0            3        0,2            3,2
    1        SUCCESS        2            1            3        0,2            3,2
    17        SUCCESS        2            1            3        0,2            3,2
    33        SUCCESS        2            1            3        0,2            3,2
    2        SUCCESS        2            2            3        0,2            3,2
    18        SUCCESS        2            2            3        0,2            3,2
    34        SUCCESS        2            2            3        0,2            3,2
    4        SUCCESS        2            4            3        0,2            3,2
    20        SUCCESS        2            4            3        0,2            3,2
    36        SUCCESS        2            4            3        0,2            3,2
    11        SUCCESS        3            11            4        0,2            4,2
```

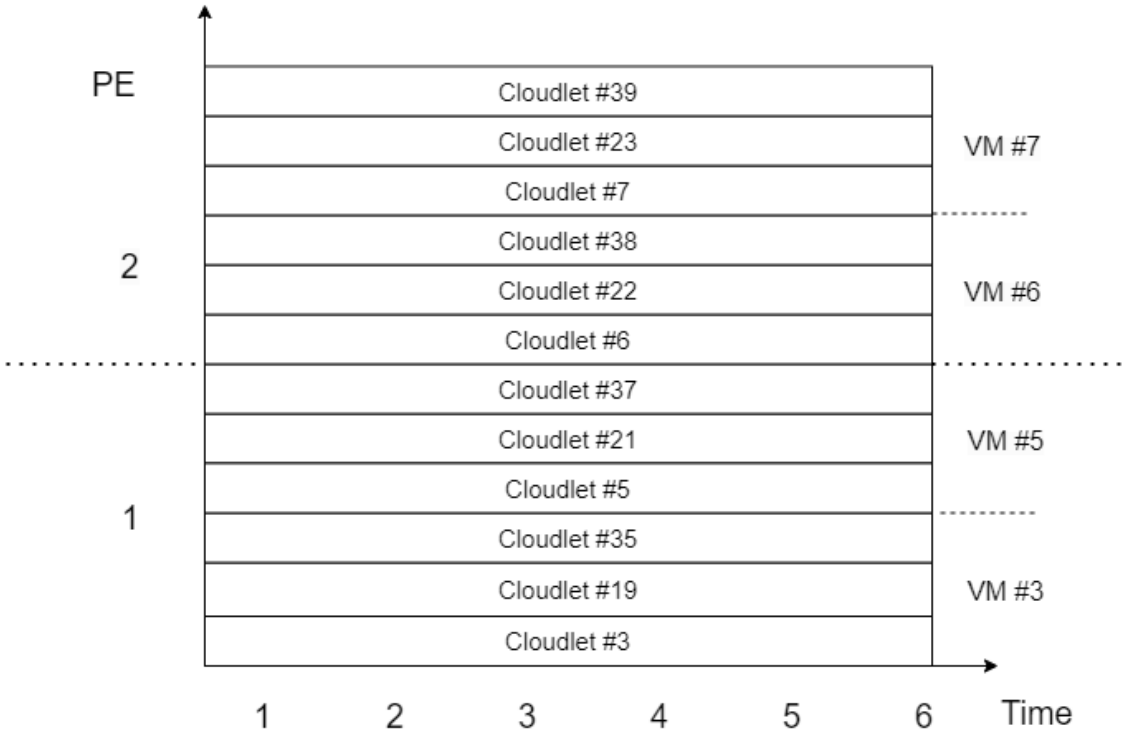
```
    27          SUCCESS          3              11              4          0,2              4,2
    13          SUCCESS          3              13              4          0,2              4,2
    29          SUCCESS          3              13              4          0,2              4,2
    14          SUCCESS          3              14              4          0,2              4,2
    30          SUCCESS          3              14              4          0,2              4,2
    15          SUCCESS          3              15              4          0,2              4,2
    31          SUCCESS          3              15              4          0,2              4,2
    3         SUCCESS          2              3               6          0,2              6,2
    19          SUCCESS          2              3               6          0,2              6,2
    35          SUCCESS          2              3               6          0,2              6,2
    5         SUCCESS          2              5               6          0,2              6,2
    21          SUCCESS          2              5               6          0,2              6,2
    37          SUCCESS          2              5               6          0,2              6,2
    6         SUCCESS          2              6               6          0,2              6,2
    22          SUCCESS          2              6               6          0,2              6,2
    38          SUCCESS          2              6               6          0,2              6,2
    7         SUCCESS          2              7               6          0,2              6,2
    23          SUCCESS          2              7               6          0,2              6,2
    39          SUCCESS          2              7               6          0,2              6,2
CloudSimExample6 finished!
```

Οι χρόνοι για τα Cloudlets των VM που έχουν ανατεθεί στους Host #0 των Datacenter #2 και #3, παραμένουν ίδιοι όπως έχει τεκμηριωθεί και στο ερώτημα 9 για την πολιτική **TC-TV**. Παρατηρούμε, όμως ότι οι χρόνοι εκτέλεσης για τα Cloudlets των VM που έχουν ανατεθεί στους Host #1 των Datacenter #2 και #3 έχουν αυξηθεί στα 6.2 second. Αυτό οφείλεται στην αυξημένη κατανομή των Cloudlets στα VMs και των VMs στου πυρήνες του Host #2, ο οποίος φιλοξενεί περισσότερα VMs απ' όσα στοιχεία επεξεργασίας διαθέτει (4 VMs και 2 PEs).

Ας πάρουμε για παράδειγμα τον Host #1 του Datacenter #2.

Ο Host #1 είναι μία φυσική 2-πύρηνη μηχανή με απόδοση 1000 MIPS ανά πυρήνα. Φιλοξενεί συνολικά 4 VMs που είναι 1-πύρηνοι με απόδοση 1000 MIPS εκάστως. Με βάση την πολιτική **First-Fit**, στα VM #3, #5, #6, #7 έχουν ανατεθεί 3 Cloudlets μεγέθους 1000 MI το καθένα.

Με την πολιτική δρομολόγησης **Time-Shared Cloudlet Scheduling** και **Time-Shared VM Scheduling over-subscription**, τα Cloudlets εκτελούνται ταυτόχρονα στα VMs, και τα VMs χρησιμοποιούν παράλληλα τον ίδιο πυρήνα του Host, όπως φαίνεται και στην **Εικόνα Α.4.**



Εικόνα Α.4 Time-Shared Cloudlet Scheduling και Time-Shared VM Scheduling με over-subscription στον Host #1 του Datacenter #2

Β. Πραγματοποιήστε τις ελάχιστες απαιτούμενες αλλαγές στη διατιθέμενη υποδομή (hosts, PEs και χαρακτηριστικά τους) ώστε να μπορούν να τρέξουν (χωρίς over-subscription) όλες οι VMs. Δώστε τρεις εκδοχές, (α) μία χωρίς να αυξήσετε τον αριθμό hosts και PEs (αυξάνοντας δηλαδή μόνο τα χαρακτηριστικά τους), (β) μία αυξάνοντας τον αριθμό των PEs (χωρίς να αυξήσετε τον αριθμό των hosts), και (γ) μία αυξάνοντας τον αριθμό των hosts. Μελετήστε πάλι το output, δείτε αν υπάρχουν ή όχι διαφορές, και εξηγείστε το γιατί.

Προκειμένου να μπορούν να τρέξουν και οι 20 VM που δημιουργούνται, στους Hosts των Datacenters υπάρχουν 3 εκδοχές όπου πραγματοποιούμε ελάχιστες απαιτούμενες αλλαγές στη διατιθέμενη υποδομή (Hosts, PEs και χαρακτηριστικά τους).

(α) αλλαγή στα χαρακτηριστικά των Hosts και PEs

Οι αλλαγές που προχωρήσαμε στα χαρακτηριστικά των hosts και PEs ήταν οι εξής:

1. Διπλασιάσαμε την απόδοση των PEs από 1000 MIPS στα 2000 MIPS
2. Διπλασιάσαμε την μνήμη RAM του κάθε Host από 2048 MB (2 GB) στα 4096 MB (4 GB)

CloudSimExample6/createDatacenter(String name) : Datacenter

```
int mips = 2000;

            // 3. Create PEs and add these into the list.
            //for a quad-core machine, a list of 4 PEs is required:
            peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating
            peList1.add(new Pe(1, new PeProvisionerSimple(mips)));
            peList1.add(new Pe(2, new PeProvisionerSimple(mips)));
            peList1.add(new Pe(3, new PeProvisionerSimple(mips)));

            //Another list, for a dual-core machine
            List<Pe> peList2 = new ArrayList<Pe>();

            peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
            peList2.add(new Pe(1, new PeProvisionerSimple(mips)));

            //4. Create Hosts with its id and list of PEs and add them to the list of
machines
            int hostId=0;
            int ram = 4096; //host memory (MB)
            long storage = 1000000; //host storage
            int bw = 10000;
```

Ωστόσο, όπως απαντάται και στο ερώτημα 5 δεν επαρκεί μόνο η αύξηση της RAM και της χωρητικότητας του επεξεργαστή (MIPS), αλλά και στα υπόλοιπα χαρακτηριστικά όπως η χωρητικότητα (storage) και το εύρος ζώνης δικτύου (bw).

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες (για επεξήγηση των στηλών προσομοίωσης, βλ. ερώτημα 10)

CloudSimExample6/Output

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish
Time
    0        SUCCESS        2            0            2        0,2            2,2
    20        SUCCESS        2            0            2        0,2            2,2
    1        SUCCESS        2            1            2        0,2            2,2
    21        SUCCESS        2            1            2        0,2            2,2
    2        SUCCESS        2            2            2        0,2            2,2
    22        SUCCESS        2            2            2        0,2            2,2
    4        SUCCESS        2            4            2        0,2            2,2
    24        SUCCESS        2            4            2        0,2            2,2
    6        SUCCESS        2            6            2        0,2            2,2
    26        SUCCESS        2            6            2        0,2            2,2
    8        SUCCESS        2            8            2        0,2            2,2
    28        SUCCESS        2            8            2        0,2            2,2
    10        SUCCESS        2            10            2        0,2            2,2
    30        SUCCESS        2            10            2        0,2            2,2
    11        SUCCESS        2            11            2        0,2            2,2
    31        SUCCESS        2            11            2        0,2            2,2
    3        SUCCESS        2            3            2        0,2            2,2
    23        SUCCESS        2            3            2        0,2            2,2
    5        SUCCESS        2            5            2        0,2            2,2
    25        SUCCESS        2            5            2        0,2            2,2
    7        SUCCESS        2            7            2        0,2            2,2
    27        SUCCESS        2            7            2        0,2            2,2
    9        SUCCESS        2            9            2        0,2            2,2
    29        SUCCESS        2            9            2        0,2            2,2
    12        SUCCESS        3            12            2        0,2            2,2
    32        SUCCESS        3            12            2        0,2            2,2
    13        SUCCESS        3            13            2        0,2            2,2
    33        SUCCESS        3            13            2        0,2            2,2
    14        SUCCESS        3            14            2        0,2            2,2
    34        SUCCESS        3            14            2        0,2            2,2
    16        SUCCESS        3            16            2        0,2            2,2
    36        SUCCESS        3            16            2        0,2            2,2
    18        SUCCESS        3            18            2        0,2            2,2
    38        SUCCESS        3            18            2        0,2            2,2
    15        SUCCESS        3            15            2        0,2            2,2
    35        SUCCESS        3            15            2        0,2            2,2
    17        SUCCESS        3            17            2        0,2            2,2
    37        SUCCESS        3            17            2        0,2            2,2
    19        SUCCESS        3            19            2        0,2            2,2
    39        SUCCESS        3            19            2        0,2            2,2
CloudSimExample6 finished!
```

Με αφορμή την κατανομή των Cloudlets στα VM με την μέθοδο **First-Fit** (βλ. ερώτημα 8) και τις μεθόδους δρομολόγησης **Time-Shared Cloudlet** και **Time-Shared VM**, το κάθε VM διαθέτει από 2 Cloudlets των 1000 MI. Τα VM εκτελούν παράλληλα τα Cloudlets και διαμοιράζονται το PE του Host που τους φιλοξενεί, και με άλλα VM. Συνεπώς με παράλληλη εκτέλεση η απόδοση του PE διαμοιράζεται για την εκτέλεση των Cloudlets και με δεδομένο ότι το

κάθε Cloudlet χρειάζεται 1 second για την εκτέλεση του στο PE του VM, και το κάθε VM διαθέτει από 2 Cloudlet, ο χρόνος εκτέλεσης του κάθε Cloudlet δικαιολογείται ορθά στα 2 second.

(β) αλλαγή στο πλήθος των PEs

Οι αλλαγές που προχωρήσαμε στο πλήθος των PEs ήταν στην αύξηση του αντίστοιχου πλήθους του Host #1 από 2 PEs στα 4 PEs.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
// 3. Create PEs and add these into the list.
            //for a quad-core machine, a list of 4 PEs is required:
            peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating
            peList1.add(new Pe(1, new PeProvisionerSimple(mips)));
            peList1.add(new Pe(2, new PeProvisionerSimple(mips)));
            peList1.add(new Pe(3, new PeProvisionerSimple(mips)));


            //Another list, for a dual-core machine
            List<Pe> peList2 = new ArrayList<Pe>();

            peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
            peList2.add(new Pe(1, new PeProvisionerSimple(mips)));
            peList2.add(new Pe(2, new PeProvisionerSimple(mips)));
            peList2.add(new Pe(3, new PeProvisionerSimple(mips)));
```

Ωστόσο, δεν καθιστά εφικτό να τρέξουν όλες οι VM στους Host, όσο και να αυξήσουμε τον αριθμό των PEs και αυτό οφείλεται στην περιορισμένη μνήμη RAM (2048 MB). Επομένως, το μέγιστο που μπορούμε να πετύχουμε είναι 16/20 VM να τρέχουν στους Host.

CloudSimExample6/Output

```
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
0.1: Broker: VM #6 has been created in Datacenter #2, Host #0
0.1: Broker: VM #7 has been created in Datacenter #2, Host #1
...
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #1
0.2: Broker: VM #10 has been created in Datacenter #3, Host #0
0.2: Broker: VM #11 has been created in Datacenter #3, Host #1
0.2: Broker: VM #12 has been created in Datacenter #3, Host #0
0.2: Broker: VM #13 has been created in Datacenter #3, Host #1
0.2: Broker: VM #14 has been created in Datacenter #3, Host #0
0.2: Broker: VM #15 has been created in Datacenter #3, Host #1
```

Έστω, ότι ο κάθε Host διαθέτει επαρκή χωρητικότητα επεξεργασίας για να φιλοξενήσει πολλά VM. Το κάθε VM διαθέτει από 512 MB μνήμη RAM και ο κάθε Host από 2048 MB, συνεπώς, ο κάθε Host μπορεί να φιλοξενήσει το πολύ 4 VM. Συνολικά, δημιουργούνται 4 Host, 2

για κάθε Datacenter. Επομένως, ο περιορισμός στη μνήμη RAM οδηγούν στο συμπέρασμα ότι ο μέγιστος αριθμός των φιλοξενούμενων VM δεν μπορεί να υπερβεί τα 16, όσα PE κι αν προσθέσουμε.

CloudSimExample6/Output

```
[VmScheduler.vmCreate] Allocation of VM #16 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #16 to Host #1 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #17 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #17 to Host #1 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #18 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #18 to Host #1 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #19 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #19 to Host #1 failed by RAM
```

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες (για επεξήγηση των στηλών προσομοίωσης, βλ. ερώτημα 10)

CloudSimExample6/Output

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish
Time
    8        SUCCESS        3            8            2        0,2            2,2
    24        SUCCESS        3            8            2        0,2            2,2
    10        SUCCESS        3            10            2        0,2            2,2
    26        SUCCESS        3            10            2        0,2            2,2
    12        SUCCESS        3            12            2        0,2            2,2
    28        SUCCESS        3            12            2        0,2            2,2
    14        SUCCESS        3            14            2        0,2            2,2
    30        SUCCESS        3            14            2        0,2            2,2
    9        SUCCESS        3            9            2        0,2            2,2
    25        SUCCESS        3            9            2        0,2            2,2
    11        SUCCESS        3            11            2        0,2            2,2
    27        SUCCESS        3            11            2        0,2            2,2
    13        SUCCESS        3            13            2        0,2            2,2
    29        SUCCESS        3            13            2        0,2            2,2
    15        SUCCESS        3            15            2        0,2            2,2
    31        SUCCESS        3            15            2        0,2            2,2
    0        SUCCESS        2            0            3        0,2            3,2
    16        SUCCESS        2            0            3        0,2            3,2
    32        SUCCESS        2            0            3        0,2            3,2
    2        SUCCESS        2            2            3        0,2            3,2
    18        SUCCESS        2            2            3        0,2            3,2
    34        SUCCESS        2            2            3        0,2            3,2
    4        SUCCESS        2            4            3        0,2            3,2
    20        SUCCESS        2            4            3        0,2            3,2
    36        SUCCESS        2            4            3        0,2            3,2
    6        SUCCESS        2            6            3        0,2            3,2
    22        SUCCESS        2            6            3        0,2            3,2
    38        SUCCESS        2            6            3        0,2            3,2
    1        SUCCESS        2            1            3        0,2            3,2
    17        SUCCESS        2            1            3        0,2            3,2
    33        SUCCESS        2            1            3        0,2            3,2
    3        SUCCESS        2            3            3        0,2            3,2
```

```
    19        SUCCESS        2            3            3        0,2            3,2
    35        SUCCESS        2            3            3        0,2            3,2
    5        SUCCESS        2            5            3        0,2            3,2
    21        SUCCESS        2            5            3        0,2            3,2
    37        SUCCESS        2            5            3        0,2            3,2
    7        SUCCESS        2            7            3        0,2            3,2
    23        SUCCESS        2            7            3        0,2            3,2
    39        SUCCESS        2            7            3        0,2            3,2
CloudSimExample6 finished!
```

Με αφορμή την κατανομή των Cloudlets στα VM με την μέθοδο **First-Fit** (βλ. ερώτημα 8) και τις μεθόδους δρομολόγησης **Time-Shared Cloudlet** και **Time-Shared VM**, το κάθε VM διαθέτει από 2 Cloudlets των 1000 MI εκτός 4 που διαθέτουν από 3 Cloudlet. Τα VM εκτελούν παράλληλα τα Cloudlets και διαμοιράζονται το PE του Host που τους φιλοξενεί, και με άλλα VM. Συνεπώς με παράλληλη εκτέλεση η απόδοση του PE διαμοιράζεται για την εκτέλεση των Cloudlets και με δεδομένο ότι το κάθε Cloudlet χρειάζεται 1 second για την εκτέλεση του στο PE του VM, και το κάθε VM που διαθέτει από 2 Cloudlet, ο χρόνος εκτέλεσης του κάθε Cloudlet δικαιολογείται ορθά στα 2 second και αντίστοιχα 3 second στα VM που διαθέτουν από 3 Cloudlet.

(γ) αλλαγή στο πλήθος των Hosts

Οι αλλαγές που πραγματοποιήσαμε ήταν στην αύξηση των Host από 2 σε 3. Αξίζει να σημειωθεί ότι ο 3ος Host διαθέτει 4 PEs των 1000 MIPS. Έτσι, λοιπόν, το κάθε Datacenter διαθέτει από 3 Host.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
hostList.add(
                    new Host(
                            hostId,
                            new RamProvisionerSimple(ram),
                            new BwProvisionerSimple(bw),
                            storage,
                            peList1,
                            new VmSchedulerTimeShared(peList1)
                    )
            ); // This is our first machine

            hostId++;

            hostList.add(
                    new Host(
                            hostId,
                            new RamProvisionerSimple(ram),
                            new BwProvisionerSimple(bw),
                            storage,
                            peList2,
                            new VmSchedulerTimeShared(peList2)
                    )
            ); // Second machine

            hostId++;

            hostList.add(
```

```
                    new Host(
                            hostId,
                            new RamProvisionerSimple(ram),
                            new BwProvisionerSimple(bw),
                            storage,
                            peList1,
                            new VmSchedulerTimeShared(peList1)
                    )
            ); // This is our third machine
```

Αξίζει, να σημειωθεί επίσης ότι η αύξηση του πλήθους των Host δεν επαρκούσε στην περίπτωση που ο 3ος Host είχε στην κατοχή του 2 PEs των 1000 MIPS αντί για 4, πράγμα που επισημάνει την άμεση εξάρτηση των Host με τα PEs.

CloudSimExample6/Output

```
[VmScheduler.vmCreate] Allocation of VM #6 to Host #2 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #7 to Host #2 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #8 to Host #2 failed by MIPS
```

Το τελικό μέρος των αποτελεσμάτων της προσομοίωσης του παραδείγματος **CloudSimExample6** απεικονίζει έναν πίνακα με τις εξής στήλες (για επεξήγηση των στηλών προσομοίωσης, βλ. ερώτημα 10)

CloudSimExample6/Output

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish
Time
    0        SUCCESS        2            0            2        0,2            2,2
    20        SUCCESS        2            0            2        0,2            2,2
    2        SUCCESS        2            2            2        0,2            2,2
    22        SUCCESS        2            2            2        0,2            2,2
    4        SUCCESS        2            4            2        0,2            2,2
    24        SUCCESS        2            4            2        0,2            2,2
    7        SUCCESS        2            7            2        0,2            2,2
    27        SUCCESS        2            7            2        0,2            2,2
    5        SUCCESS        2            5            2        0,2            2,2
    25        SUCCESS        2            5            2        0,2            2,2
    8        SUCCESS        2            8            2        0,2            2,2
    28        SUCCESS        2            8            2        0,2            2,2
    1        SUCCESS        2            1            2        0,2            2,2
    21        SUCCESS        2            1            2        0,2            2,2
    3        SUCCESS        2            3            2        0,2            2,2
    23        SUCCESS        2            3            2        0,2            2,2
    6        SUCCESS        2            6            2        0,2            2,2
    26        SUCCESS        2            6            2        0,2            2,2
    9        SUCCESS        2            9            2        0,2            2,2
    29        SUCCESS        2            9            2        0,2            2,2
    10        SUCCESS        3            10            2        0,2            2,2
    30        SUCCESS        3            10            2        0,2            2,2
    12        SUCCESS        3            12            2        0,2            2,2
    32        SUCCESS        3            12            2        0,2            2,2
    14        SUCCESS        3            14            2        0,2            2,2
    34        SUCCESS        3            14            2        0,2            2,2
```

```
    17          SUCCESS         3           17          2           0,2         2,2
    37          SUCCESS         3           17          2           0,2         2,2
    15          SUCCESS         3           15          2           0,2         2,2
    35          SUCCESS         3           15          2           0,2         2,2
    18          SUCCESS         3           18          2           0,2         2,2
    38          SUCCESS         3           18          2           0,2         2,2
    11          SUCCESS         3           11          2           0,2         2,2
    31          SUCCESS         3           11          2           0,2         2,2
    13          SUCCESS         3           13          2           0,2         2,2
    33          SUCCESS         3           13          2           0,2         2,2
    16          SUCCESS         3           16          2           0,2         2,2
    36          SUCCESS         3           16          2           0,2         2,2
    19          SUCCESS         3           19          2           0,2         2,2
    39          SUCCESS         3           19          2           0,2         2,2
CloudSimExample6 finished!
```

Το output της προσομοίωσης παραμένει το ίδιο με την περίπτωση α.

Γ. Σε ποια σημεία του κώδικα (υλοποίησης του simulator) και πώς θα επεμβαίνατε (α) για να αλλάξετε την πολιτική / αλγόριθμο απόφασης για τα 5. και 8., και (β) για να προσθέσετε κάποια άλλη δικιά σας πολιτική για τα 9(α) και 9(β); Ειδικότερα όσον αφορά το 5. (την πολιτική που ακολουθείται δηλαδή για την ανάθεση των VMs στους Hosts), εξηγήστε με περισσότερη λεπτομέρεια πώς θα υλοποιούσατε μία άλλη πολιτική της επιλογής σας – αναζητήστε και επιλέξτε μία άλλη συγκεκριμένη πολιτική (συμβουλευτείτε μεταξύ άλλων και τα επισυναπτόμενα links και papers – βλ. folder 'VM Allocation Policies'), περιγράψτε τη σύντομα, και προσπαθήστε στη συνέχεια να την υλοποιήσετε.

(α) αλλαγή της πολιτικής ανάθεσης των VM στους Hosts, και των Cloudlets στα VM

Η πολιτική απόφασης που αναθέτει τα VM στους Hosts υλοποιείται στην κλάση **VmAllocationPolicySimple.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
// 6. Finally, we need to create a PowerDatacenter object.
            Datacenter datacenter = null;
            try {
                  datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
            } catch (Exception e) {
                  e.printStackTrace();
            }
```

Αν επιθυμούσαμε να αλλάξουμε την πολιτική αυτή θα δημιουργούσαμε μία νέα κλάση που θα είχε μέσα μία άλλη πολιτική απόφασης για την ανάθεση των VM στους Hosts. Η νέα κλάση θα

κληρονομούσε την κλάση **VmAllocationPolicy** που χρησιμοποιεί τη μέθοδο **allocateHostForVm()** και έτσι θα είχαμε την δυνατότητα να υλοποιήσουμε την νέα ιδέα της κατανομής των VM στους Host, σ' αυτή τη μέθοδο.

VmAllocationPolicySimple/allocateHostForVm(Vm vm) : boolean

```
public boolean allocateHostForVm(Vm vm) {
		int requiredPes = vm.getNumberOfPes();
		boolean result = false;
		int tries = 0;
		List<Integer> freePesTmp = new ArrayList<Integer>();
		for (Integer freePes : getFreePes()) {
			freePesTmp.add(freePes);
		}

		if (!getVmTable().containsKey(vm.getUid())) { // if this vm was not created
			do {// we still trying until we find a host or until we try all of
them
				int moreFree = Integer.MIN_VALUE;
				int idx = -1;

				// we want the host with less pes in use
				for (int i = 0; i < freePesTmp.size(); i++) {
					if (freePesTmp.get(i) > moreFree) {
						moreFree = freePesTmp.get(i);
						idx = i;
					}
				}

				Host host = getHostList().get(idx);
				result = host.vmCreate(vm);

				if (result) { // if vm were succesfully created in the host
					getVmTable().put(vm.getUid(), host);
					getUsedPes().put(vm.getUid(), requiredPes);
					getFreePes().set(idx, getFreePes().get(idx) -
requiredPes);

					result = true;
					break;
				} else {
					freePesTmp.set(idx, Integer.MIN_VALUE);
				}
				tries++;
			} while (!result && tries < getFreePes().size());

		}

		return result;
	}
```

Η πολιτική απόφασης που αναθέτει τα Cloudlets στα VM υλοποιείται στην κλάση **DatacenterBroker.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudsimExample6/main(String[] args) : void

```
//Third step: Create Broker
                    DatacenterBroker broker = createBroker();
                    int brokerId = broker.getId();

                    //Fourth step: Create VMs and Cloudlets and send them to broker
                    vmlist = createVM(brokerId,20); //creating 20 vms
                    cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets

                    broker.submitVmList(vmlist);
                    broker.submitCloudletList(cloudletList);
```

Αν επιθυμούσαμε να αλλάξουμε την πολιτική ανάθεσης των Cloudlets στα VM θα επεμβέναμε στην μέθοδο **submitCloudlets()** της κλάσης **DatacenterBroker**. Η οντότητα broker είναι μία οντότητα που διαχειρίζεται την δημιουργία των VM, την ανάθεση των Cloudlets στα VM και την καταστροφή των VM μετά την ολοκλήρωση της προσομοίωσης.

DatacenterBroker/submitCloudlets() : void

```
protected void submitCloudlets() {
            int vmIndex = 0;
            List<Cloudlet> successfullySubmitted = new ArrayList<Cloudlet>();
            for (Cloudlet cloudlet : getCloudletList()) {
                  Vm vm;
                  // if user didn't bind this cloudlet and it has not been executed
yet
                  if (cloudlet.getVmId() == -1) {
                        vm = getVmsCreatedList().get(vmIndex);
                  } else { // submit to the specific vm
                        vm = VmList.getById(getVmsCreatedList(), cloudlet.getVmId());
                        if (vm == null) { // vm was not created
                              if(!Log.isDisabled()) {
                                  Log.printConcatLine(CloudSim.clock(), ": ",
getName(), ": Postponing execution of cloudlet ",
                                                cloudlet.getCloudletId(), ": bount VM not
available");
                              }
                              continue;
                        }
                  }

                  if (!Log.isDisabled()) {
                      Log.printConcatLine(CloudSim.clock(), ": ", getName(), ":
Sending cloudlet ",
                              cloudlet.getCloudletId(), " to VM #", vm.getId());
                  }

                  cloudlet.setVmId(vm.getId());
                  sendNow(getVmsToDatacentersMap().get(vm.getId()),
CloudSimTags.CLOUDLET_SUBMIT, cloudlet);
                  cloudletsSubmitted++;
                  vmIndex = (vmIndex + 1) % getVmsCreatedList().size();
```

```
                    getCloudletSubmittedList().add(cloudlet);
                    successfullySubmitted.add(cloudlet);
              }

              // remove submitted cloudlets from waiting list
              getCloudletList().removeAll(successfullySubmitted);
        }
```

(β) προσθήκη νέας πολιτικής δρομολόγησης των VM στα PE των Host και των Cloudlets στα PE των VM

Η πολιτική δρομολόγησης που χρονοπρογραμματίζει τα VM στα PE των Host, υλοποιείται στην κλάση **VmScheduler.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createDatacenter(String name) : Datacenter

```
hostList.add(
                    new Host(
                          hostId,
                          new RamProvisionerSimple(ram),
                          new BwProvisionerSimple(bw),
                          storage,
                          peList1,
                          new VmSchedulerTimeShared(peList1)
                    )
              ); // This is our first machine
```

Αν επιθυμούσαμε την προσθήκη νέας πολιτικής δρομολόγησης θα δημιουργούσαμε μια νέα κλάση που θα κληρονομούσε την **VmScheduler** που χρησιμοποιεί τη μέθοδο **allocatePesForVm()** και έτσι θα είχαμε την δυνατότητα να υλοποιήσουμε την νέα ιδέα δρομολόγησης των VM στα PE των Host, σ' αυτή τη μέθοδο.

VmSchedulerTimeShared/allocatePesForVm(String vmUid, List<Double> mipsSharedRequested) : boolean

```
protected boolean allocatePesForVm(String vmUid, List<Double> mipsShareRequested) {
              double totalRequestedMips = 0;
              double peMips = getPeCapacity();
              for (Double mips : mipsShareRequested) {
                    // each virtual PE of a VM must require not more than the capacity
of a physical PE
                    if (mips > peMips) {
                          return false;
                    }
                    totalRequestedMips += mips;
              }

              // This scheduler does not allow over-subscription
              if (getAvailableMips() < totalRequestedMips) {
                    return false;
              }

              getMipsMapRequested().put(vmUid, mipsShareRequested);
              setPesInUse(getPesInUse() + mipsShareRequested.size());
```

```
		if (getVmsMigratingIn().contains(vmUid)) {
			// the destination host only experience 10% of the migrating VM's
MIPS
			totalRequestedMips *= 0.1;
		}

		List<Double> mipsShareAllocated = new ArrayList<Double>();
		for (Double mipsRequested : mipsShareRequested) {
			if (getVmsMigratingOut().contains(vmUid)) {
				// performance degradation due to migration = 10% MIPS
				mipsRequested *= 0.9;
			} else if (getVmsMigratingIn().contains(vmUid)) {
				// the destination host only experience 10% of the migrating
VM's MIPS
				mipsRequested *= 0.1;
			}
			mipsShareAllocated.add(mipsRequested);
		}

		getMipsMap().put(vmUid, mipsShareAllocated);
		setAvailableMips(getAvailableMips() - totalRequestedMips);

		return true;
	}
```

Η πολιτική δρομολόγησης που χρονοπρογραμματίζει τα Cloudlets στα PE των VM, υλοποιείται στην κλάση **CloudletScheduler.java** που βρίσκεται στο πακέτο **org.cloudbus.cloudsim**.

CloudSimExample6/createVM(int userId, int vms) : List<Vm>

```
//create VMs
		Vm[] vm = new Vm[vms];

		for(int i=0;i<vms;i++){
			vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
			//for creating a VM with a space shared scheduling policy for
cloudlets:
			//vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerSpaceShared());

			list.add(vm[i]);
		}
```

Αν επιθυμούσαμε να προσθέσουμε μια νέα πολιτική δρομολόγηση των Cloudlets στα VM θα δημιουργούσαμε μια νέα κλάση που θα κληρονομούσε την **CloudletScheduler** που χρησιμοποιεί τη μέθοδο **updateVmProcessing()** και έτσι θα είχαμε την δυνατότητα να υλοποιήσουμε την νέα ιδέα δρομολόγησης των Cloudlets στα PE των VM, σ' αυτή τη μέθοδο.

CloudletSchedulerTimeShared/updateVmProcessing(double currentTime, List<Double> mipsShare) : double

```
@Override
	public double updateVmProcessing(double currentTime, List<Double> mipsShare) {
```

```
            setCurrentMipsShare(mipsShare);
            double timeSpam = currentTime - getPreviousTime();

            for (ResCloudlet rcl : getCloudletExecList()) {
                  rcl.updateCloudletFinishedSoFar((long) (getCapacity(mipsShare) *
timeSpam * rcl.getNumberOfPes() * Consts.MILLION));
            }

            if (getCloudletExecList().size() == 0) {
                  setPreviousTime(currentTime);
                  return 0.0;
            }

            // check finished cloudlets
            double nextEvent = Double.MAX_VALUE;
            List<ResCloudlet> toRemove = new ArrayList<ResCloudlet>();
            for (ResCloudlet rcl : getCloudletExecList()) {
                  long remainingLength = rcl.getRemainingCloudletLength();
                  if (remainingLength == 0) {// finished: remove from the list
                        toRemove.add(rcl);
                        cloudletFinish(rcl);
                        continue;
                  }
            }
            getCloudletExecList().removeAll(toRemove);

            // estimate finish time of cloudlets
            for (ResCloudlet rcl : getCloudletExecList()) {
                  double estimatedFinishTime = currentTime
                              + (rcl.getRemainingCloudletLength() /
(getCapacity(mipsShare) * rcl.getNumberOfPes()));
                  if (estimatedFinishTime - currentTime <
CloudSim.getMinTimeBetweenEvents()) {
                        estimatedFinishTime = currentTime +
CloudSim.getMinTimeBetweenEvents();
                  }

                  if (estimatedFinishTime < nextEvent) {
                        nextEvent = estimatedFinishTime;
                  }
            }

            setPreviousTime(currentTime);
            return nextEvent;
      }
```

Όπως αναφέρθηκε στα προαναφερόμενα, για να αλλάξουμε την πολιτική ανάθεσης των VM στους Host, θα δημιουργήσουμε μία νέα κλάση που θα κληρονομήσει την **VmAllocationPolicy**, ώστε να επεμβούμε στην μέθοδο **allocateHostForVm()** που χειρίζεται την πολιτική για την ανάθεση των VM στους Host.

Η νέα πολιτική που θα χρησιμοποιήσουμε είναι η **Round-Robin** πολιτική. Η πολιτική αυτή αναθέτει τα VM στους Host με κυκλική σειρά. Κάθε αίτηση VM ανατίθεται στον αμέσως επόμενο κατά σειρά enity_id Host ο οποίος δέχεται το VM, εφόσον οι διαθέσιμοι πόροι του επαρκούν για την φιλοξενεία. Η πολιτική **Round-Robin** διασφαλίζει την ομοιόμορφη κατανομή των VM στους Host αποτρέποντας την υπερφόρτωση οποιουδήποτε μεμονωμένου Host.

Για την υλοποίηση της πολιτικής, προσθέσαμε αρχικά την κλάση **CircularHostList.java** στο πακέτο **org.cloudbus.cloudsim** από το αποθετήριο που βρίσκεται στον παρακάτω σύνδεσμο:

https://github.com/AnanthaRajuC/CloudSim-Round-Robin/tree/master/src/cloudsim_round_robin

Η κλάση αυτή κληρονομεί την κλάση **Iterable<T>** που δίνει την δυνατότητα στο αντικείμενο (στην περίπτωση μας ο Host) να είναι στόχος της ενισχυμένης εντολής for. Έτσι, έχοντας τους Host αποθηκευμένους σε μία συνδεδεμένη λίστα (**LinkedList<Host>**) μπορούμε να τους ανακτούμε με προσπέλαση σε κυκλική σειρά.

CircularHostList

```
public final class CircularHostList implements Iterable<Host> {

	private final List<Host> host_list = new LinkedList<Host>();

	private int ini;

	public CircularHostList(List<? extends Host> hosts) {
		this.host_list.addAll(hosts);
	}

	public boolean add(Host host) {
		return this.host_list.add(host);
	}

	public boolean remove(Host host2Remove) {
		return this.host_list.remove(host2Remove);
	}

	public Host next() {
		Host host = null;

		if (!host_list.isEmpty()) {
			int index = (this.ini++ % this.host_list.size());
			host = this.host_list.get(index);
		}

		return host;
	}
```

Στη συνέχεια, προσθέσαμε την κλάση **RoundRobinVmAllocationPolicy** στο πακέτο **org.cloudbus.cloudsim** από το αποθετήριο που βρίσκεται στον σύνδεσμο:

https://gist.github.com/alessandroleite/4598040

Η κλάση τηρεί όσα αναφέραμε στα προαναφερόμενα, δηλαδή, στην μέθοδο **allocateHostForVm()** που κληρονομεί από την κλάση **VmAllocationPolicy**, εφαρμόζει την πολιτική **Round-Robin** για την ανάθεση των VM στους Host.

RoundRobinVmAllocationPolicy/allocateHostForVm(Vm vm, Host host) : boolean

```
public boolean allocateHostForVm(Vm vm, Host host)
	{
		if (host != null && host.vmCreate(vm))
		{
```

```
                    vm_table.put(vm.getUid(), host);
                    Log.formatLine("%.4f: VM #" + vm.getId() + " has been allocated to
the host#" + host.getId() +
                                    " datacenter #" + host.getDatacenter().getId() + "(" +
host.getDatacenter().getName() + ") #",
                                    CloudSim.clock());
                    return true;
                }
                return false;
        }
```

Τα αποτελέσματα της προσομοίωσης για τους Host του Datacenter #2 είναι τα εξής:

CloudSimExample6/Output

```
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: Creation of VM #5 failed in Datacenter #2
0.1: Broker: VM #6 has been created in Datacenter #2, Host #0
```

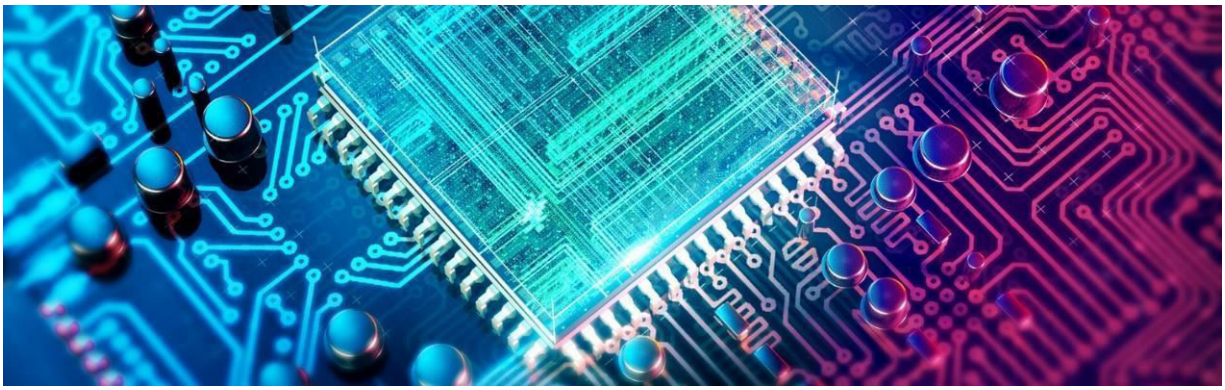
Τα αποτελέσματα της προσομοίωσης για τους Host του Datacenter #3 είναι τα εξής:

CloudSimExample6/Output

```
0.2: Broker: VM #5 has been created in Datacenter #3, Host #0
0.2: Broker: VM #7 has been created in Datacenter #3, Host #1
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #1
0.2: Broker: VM #10 has been created in Datacenter #3, Host #0
0.2: Broker: Creation of VM #11 failed in Datacenter #3
0.2: Broker: VM #12 has been created in Datacenter #3, Host #0
```

Όπως παρατηρούμε, τα VM κατανέμονται εναλλάξ στους διαθέσιμους Host του κάθε Datacenter, λόγω της κυκλικής σειράς που υιοθετεί η πολιτική **Round-Robin**. Αξίζει, να σημειωθεί ότι στην περίπτωση που κάποιος Host δεν μπορεί να εξυπηρετήσει το αίτημα του VM λόγω περιορισμένων διαθέσιμων πόρων, το αμέσως επόμενο VM στέλνει αίτημα στον αμέσως επόμενο Host κατά την κυκλική σειρά.

Σας ευχαριστώ για την προσοχή σας.



ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΕΦΟΥΣ ΚΑΙ ΥΠΗΡΕΣΙΕΣ