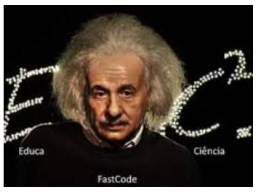


O que é CloudSim?

CloudSim é uma ferramenta poderosa e bastante utilizada no mundo da computação em nuvem.

Imagine-o como um laboratório virtual onde você pode construir e testar diferentes cenários de nuvens computacionais, sem a necessidade de um hardware real.

Em termos mais simples:

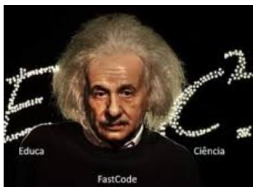
- **Simulador de Nuvens:** CloudSim permite que você simule o comportamento de uma nuvem computacional, desde a criação de máquinas virtuais até a alocação de recursos como CPU, memória e armazenamento.
- **Ferramenta de Pesquisa:** Pesquisadores e estudantes utilizam o CloudSim para desenvolver e avaliar novos algoritmos e estratégias para gerenciamento de recursos em nuvens.
- **Plataforma de Ensino:** É uma excelente ferramenta para aprender sobre os conceitos e funcionamento de nuvens computacionais.

Por que usar o CloudSim?

- **Flexibilidade:** Permite criar cenários personalizados e complexos.
- **Custo-benefício:** Evita gastos com infraestrutura física.
- **Agilidade:** Permite realizar diversas simulações em pouco tempo.
- **Facilidade de uso:** Possui uma interface intuitiva e documentação completa.

Um exemplo simples de uso: Imagine que você quer comparar o desempenho de dois algoritmos diferentes para alocação de máquinas virtuais em uma nuvem.

1. **Modelo da Nuvem:** No CloudSim, você modelaria sua nuvem definindo o número de data centers, a capacidade de processamento de cada servidor, a quantidade de memória disponível e outros recursos.



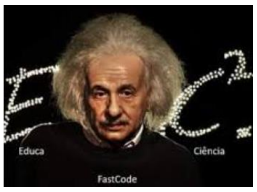
2. **Criação de Carga de Trabalho:** Em seguida, você criaria uma carga de trabalho, simulando a chegada de novas máquinas virtuais ao longo do tempo, com diferentes requisitos de recursos.
3. **Implementação dos Algoritmos:** Você implementaria os dois algoritmos de alocação que deseja comparar, indicando como cada um decide qual máquina virtual alocar em qual servidor.
4. **Simulação:** Ao executar a simulação, o CloudSim irá simular a execução dos algoritmos, considerando a carga de trabalho e os recursos disponíveis na nuvem.
5. **Análise de Resultados:** Por fim, você analisaria os resultados da simulação, comparando o tempo de resposta das máquinas virtuais, a taxa de utilização dos recursos e outros indicadores de desempenho, para determinar qual algoritmo apresenta o melhor desempenho para o cenário em questão.

   O CloudSim é uma ferramenta indispensável para quem trabalha com computação em nuvem, seja para pesquisa, desenvolvimento ou ensino. Ele permite experimentar e avaliar diferentes soluções de forma rápida e eficiente, contribuindo para o avanço da área.

   Entendendo a Complexidade - é projetado para simular sistemas complexos, não apenas para imprimir uma mensagem na tela.

- **Nível de Abstração:** O CloudSim opera em um nível de abstração mais alto, simulando recursos como data centers, máquinas virtuais, redes e etc.
- **Foco:** Seu objetivo principal é simular o comportamento de sistemas complexos, não a execução de pequenos programas.
- **Configuração:** Para iniciar uma simulação no CloudSim, é necessário configurar diversos parâmetros, como o número de data centers, a capacidade de processamento dos servidores, a carga de trabalho, etc.

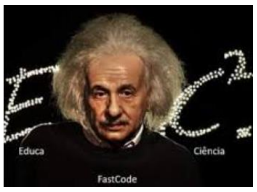
  Como exemplo podemos criar uma simulação básica no CloudSim que demonstre a criação de uma máquina virtual e a execução de uma tarefa nela.



Exemplo Básico em Java (Linguagem padrão do CloudSim):

Java

```
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

import java.util.ArrayList;
import java.util.List;

public class SimpleCloudSimExample {
    public static void main(String[] args) {
        try {
            // Inicializa a simulação
            int num_user = 1;   // Número de usuários
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false;  // Se verdadeiro, imprime
informações detalhadas
            CloudSim.init(calendar.getTime(),   2,   2,   num_user,
trace_flag);

            // Cria um datacenter
            Datacenter                datacenter0                =
createDatacenter("Datacenter_0");

            // Cria um broker
            DatacenterBroker broker = createBroker();
            int brokerId = broker.getId();

            // Cria uma máquina virtual
            Vm vm1 = new Vm(1, 2000, 1024, 1000, "vm1");
            List<Vm> vmlist = new ArrayList<Vm>();
            vmlist.add(vm1);

            // Submete as VMs ao broker
            broker.submitVmList(vmlist);

            // Cria uma Cloudlet (tarefa) para a VM
            Cloudlet cloudlet1 = new Cloudlet(1, 2000, 1, 2, 0);
            cloudlet1.setUserId(brokerId);
            List<Cloudlet> cloudletlist = new ArrayList<Cloudlet>();
            cloudletlist.add(cloudlet1);
```



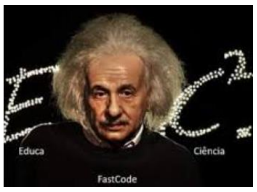

```
            // Submete as Cloudlets ao broker
            broker.submitCloudletList(cloudletlist);

            // Começa a simulação
            CloudSim.startSimulation();

            // Finaliza a simulação
            CloudSim.stopSimulation();

            // Imprime o resultado da simulação
            List<Cloudlet>              receivedList              =
broker.getCloudletReceivedList();

            //  ...  (Código  para  imprimir  informações  sobre  as
Cloudlets)

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // ... (Métodos para criar datacenter, broker, etc.)
}
```

O que esse código faz?

1. **Inicializa a simulação:** Define parâmetros como o número de usuários e o tempo de início da simulação.
2. **Cria um datacenter:** Define a capacidade de processamento, memória e outros recursos do datacenter.
3. **Cria um broker:** Representa um usuário que submete VMs e tarefas ao datacenter.
4. **Cria uma máquina virtual:** Define a quantidade de memória, CPU e outros recursos da VM.
5. **Cria uma Cloudlet:** Representa uma tarefa a ser executada na VM.
6. **Submete VMs e Cloudlets:** Envia as VMs e tarefas para o broker.
7. **Inicia a simulação:** Executa a simulação.
8. **Finaliza a simulação:** Encerra a simulação e coleta os resultados.

Importante:

CloudSim simulará a execução da tarefa, mas não imprimirá nada na tela.



- **Complexidade:** Este é um exemplo muito básico. Para simulações mais complexas, você precisará configurar diversos outros parâmetros e utilizar funcionalidades mais avançadas do CloudSim.
- **Personalização:** Você pode personalizar este código para simular diferentes cenários, como a criação de múltiplas VMs, a execução de tarefas em paralelo e a avaliação de diferentes algoritmos de alocação de recursos.

Próximos Passos de curiosidade:

- **Explore a documentação do CloudSim:** A documentação oficial oferece exemplos mais complexos e explicações detalhadas sobre as diversas funcionalidades da ferramenta.
- **Crie simulações mais complexas:** Experimente criar simulações com múltiplos data centers, diferentes tipos de máquinas virtuais e cargas de trabalho variadas.
- **Avalie diferentes algoritmos:** Utilize o CloudSim para comparar o desempenho de diferentes algoritmos de alocação de recursos e gerenciamento de energia.

Lembre-se: O CloudSim é uma ferramenta poderosa e versátil, mas exige um certo nível de conhecimento sobre simulação e computação em nuvem. Com prática e estudo, você será capaz de criar simulações cada vez mais complexas e realistas.

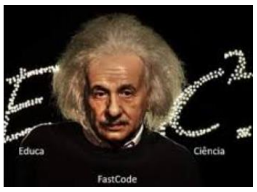
Fontes:

->https://github.com/sajan-caissa/Virtual-Machine-VM-Consolidation-using-Roulette-wheel-selection-Strategy---Genetic-Algorithm

->https://github.com/niteshdudhey/cloudsim

->https://www.scribd.com/document/233919832/Privacy-Preserving-Intermediate-Datasets-in-the-Cloud

->https://github.com/ppzgh/Energy-Efficient-Algorithms_CloudSim



Exemplos do CloudSim

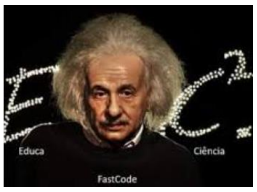
Todos os exemplos estão incluídos no pacote CloudSim.

Exemplos básicos

CloudSimExample1 – Um exemplo simples mostrando como criar um datacenter com um host e executar um cloudlet nele.
CloudSimExample2 – Um exemplo simples mostrando como criar dois datacenters com um host e uma topologia de rede cada e executar dois cloudlets neles.
CloudSimExample3 – Um exemplo simples mostrando como criar dois datacenters com um host cada e executar cloudlets de dois usuários com topologia de rede neles.
CloudSimExample4 – Um exemplo simples mostrando como criar dois datacenters com um host cada e executar dois cloudlets neles.
CloudSimExample5 – Um exemplo simples mostrando como criar dois datacenters com um host cada e executar cloudlets de dois usuários neles.
CloudSimExample6 – Um exemplo mostrando como criar simulações escaláveis.
CloudSimExample7 – Um exemplo mostrando como pausar e retomar a simulação e criar entidades de simulação (um DatacenterBroker neste exemplo) dinamicamente.
CloudSimExample8 – Um exemplo mostrando como criar entidades de simulação (um DatacenterBroker neste exemplo) em tempo de execução usando uma entidade de gerenciador global (GlobalBroker).

Exemplos de rede

NetworkExample1 – Um exemplo simples mostrando como criar um datacenter com um host e uma topologia de rede e executar um cloudlet nele.
NetworkExample2 – Um exemplo simples mostrando como criar dois datacenters com um host e uma topologia de rede cada e executar dois cloudlets neles.
NetworkExample3 – Um exemplo simples mostrando como criar dois datacenters com um host cada e executar cloudlets de dois usuários com topologia de rede neles.
NetworkExample4 – Um exemplo simples mostrando como criar um datacenter com um host e uma topologia de rede e executar um cloudlet nele. Aqui, em vez de usar um arquivo BRIE descrevendo os links, os links são inseridos no código.



Exemplos de power

NonPowerAware – Uma simulação de um data center heterogêneo sem consciência de energia: todos os hosts consomem o máximo de energia o tempo todo.
Dvfs – Uma simulação de um data center heterogêneo com consciência de energia que aplicou apenas DVFS, mas nenhuma otimização dinâmica da alocação de VM. O ajuste do consumo de energia dos hosts de acordo com a utilização da CPU está acontecendo na classe PowerDatacenter.
ThrRs – Uma simulação de um data center heterogêneo com consciência de energia que aplica a política de alocação de VM de Limite Estático (THR) e a política de seleção de VM de Seleção Aleatória (RS).
IqrMc – Uma simulação de um data center heterogêneo com consciência de energia que aplica a política de alocação de VM de Intervalo Interquartil (IQR) e a política de seleção de VM de Correlação Máxima (MC).
MadMmt – Uma simulação de um data center heterogêneo com consciência de energia que aplica a política de alocação de VM de Desvio Absoluto Mediano (MAD) e a política de seleção de VM de Tempo Mínimo de Migração (MMT).
LrMu – Uma simulação de um data center heterogêneo com consciência de energia que aplica a política de alocação de VM de Regressão Local (LR) e a política de seleção de VM de Utilização Mínima (MU).

Exemplos de contêineres

ContainerCloudSimExample1.java – Um exemplo inicial sobre o uso de simulação de contêiner.
ContainerInitialPlacementTest.java – Um exemplo explorando o problema de posicionamento inicial de contêineres.
ContainerOverbooking.java – Um exemplo explorando overbooking de contêineres.
ContainerSelectionTest.java – Um exemplo explorando a seleção de contêineres.

Vamos ver melhor um exemplo:

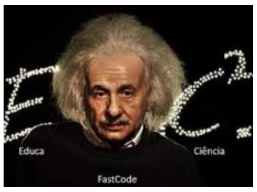
Exemplo de rede1.java

https://github.com/Cloudslab/cloudsim/blob/master/modules/cloudsim-examples/src/main/java/org/cloudbus/cloudsim/examples/network/NetworkExample1.java

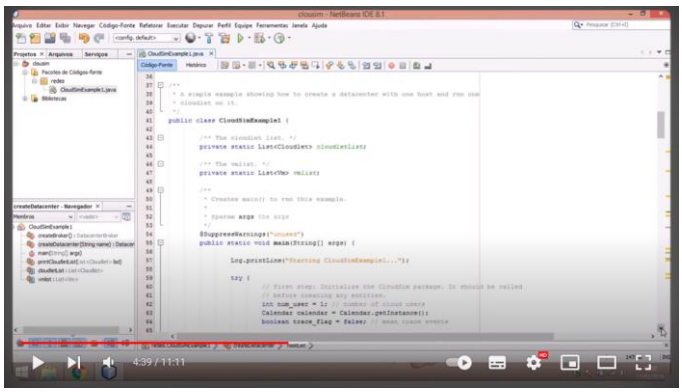
#01 COMPUTAÇÃO EM NUVEM BAIXANDO OS ARQUIVOS
https://www.youtube.com/watch?v=2SueYfo3Z5Y

#02 COMPUTAÇÃO EM NUVEM CONFIGURANDO ARQUIVOS
https://www.youtube.com/watch?v=2RjhD1TDygI&list=PLkBAC-eFXaswOk21xNJT_bHg9JcnTohb4&index=6

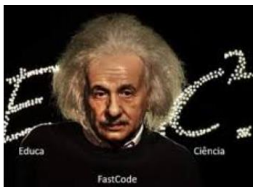
#03 COMPUTAÇÃO EM NUVEM RODANDO O EXEMPLO
https://www.youtube.com/watch?v=c9tddqDBRzk&list=PLkBAC-eFXaswOk21xNJT_bHg9JcnTohb4&index=7



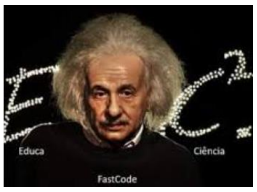
Configuração e primeiro exemplo no CloudSim
https://www.youtube.com/watch?v=XlG8pT_4Cls



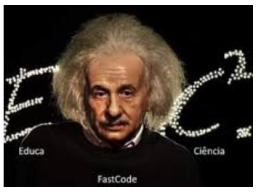
```
package educaciencia.cloud.sim;

/**
 * ********* EDUCACIENCIA FASTCODE *********
 * ********* CLOUDSIM ***********************
 * ********* DEMO EXEMPLO *******************
 * */
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.NetworkTopology;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

/**
 * A simple example showing how to create a datacenter with one host and a
 * network topology and and run one cloudlet on it.
 */
public class TesteCloudSim_run {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example
	 */
	public static void main(String[] args) {

		Log.printLine("Starting NetworkExample1...");

		try {
			// First step: Initialize the CloudSim package. It should be called
			// before creating any entities.
			int num_user = 1; // number of cloud users
```




```
                        Calendar calendar = Calendar.getInstance();
                        boolean trace_flag = false; // mean trace events

                        // Initialize the CloudSim library
                        CloudSim.init(num_user, calendar, trace_flag);

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

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

                        // Fourth step: Create one virtual machine
                        vmlist = new ArrayList<>();

                        // VM description
                        int vmid = 0;
                        int mips = 250;
                        long size = 10000; // image size (MB)
                        int ram = 512; // vm memory (MB)
                        long bw = 1000;
                        int pesNumber = 1; // number of cpus
                        String vmm = "Xen"; // VMM name

                        // create VM
                        Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

                        // add the VM to the vmList
                        vmlist.add(vm1);

                        // submit vm list to the broker
                        broker.submitVmList(vmlist);

                        // Fifth step: Create one Cloudlet
                        cloudletList = new ArrayList<>();

                        // Cloudlet properties
                        int id = 0;
                        long length = 40000;
                        long fileSize = 300;
                        long outputSize = 300;
                        UtilizationModel utilizationModel = new UtilizationModelFull();

                        Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
utilizationModel,
                                        utilizationModel, utilizationModel);
                        cloudlet1.setUserId(brokerId);

                        // add the cloudlet to the list
                        cloudletList.add(cloudlet1);

                        // submit cloudlet list to the broker
                        broker.submitCloudletList(cloudletList);

                        // Sixth step: configure network
                        // load the network topology file
                        NetworkTopology.buildNetworkTopology("topology.brite");

                        // maps CloudSim entities to BRITE entities
                        // PowerDatacenter will correspond to BRITE node 0
                        int briteNode = 0;
                        NetworkTopology.mapNode(datacenter0.getId(), briteNode);

                        // Broker will correspond to BRITE node 3
                        briteNode = 3;
                        NetworkTopology.mapNode(broker.getId(), briteNode);

                        // Seventh step: Starts the simulation
                        CloudSim.startSimulation();

                        // Final step: Print results when simulation is over
                        List<Cloudlet> newList = broker.getCloudletReceivedList();

                        CloudSim.stopSimulation();

                        printCloudletList(newList);
```

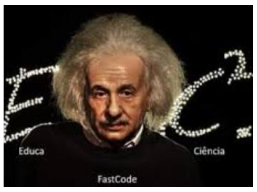


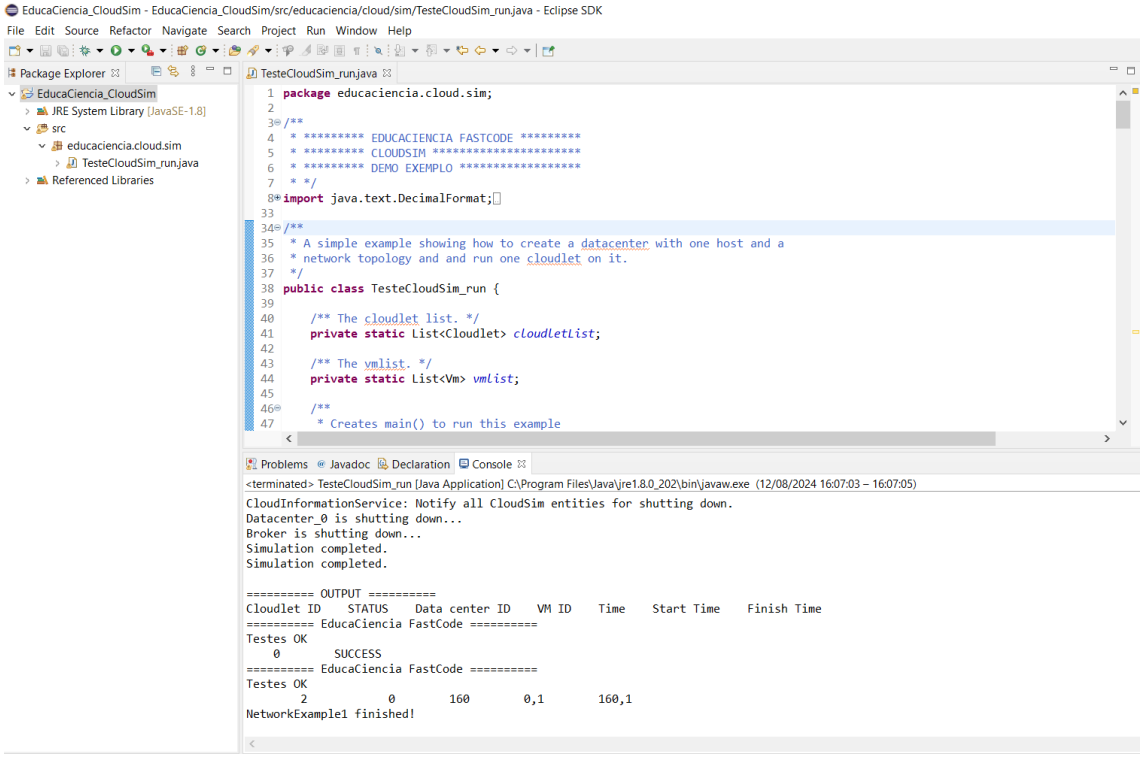

```
                Log.printLine("NetworkExample1 finished!");
            } catch (Exception e) {
                e.printStackTrace();
                Log.printLine("The simulation has been terminated due to an unexpected error");
            }
        }

        private static Datacenter createDatacenter(String name) {

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

            // 2. A Machine contains one or more PEs or CPUs/Cores.
            // In this example, it will have only one core.
            List<Pe> peList = new ArrayList<>();

            int mips = 1000;

            // 3. Create PEs and add these into a list.
            peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

            // 4. Create Host 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;

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

            // 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 = 0.001; // the cost of using storage in this resource
            double costPerBw = 0.0; // the cost of using bw in this resource
            LinkedList<Storage> storageList = new LinkedList<>(); // we are not adding SAN devices by
now

            DatacenterCharacteristics characteristics = new DatacenterCharacteristics(arch, os, vmm,
hostList, time_zone,
                    cost, costPerMem, costPerStorage, costPerBw);

            // 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();
            }

            return datacenter;
        }

        // We strongly encourage users to develop their own broker policies, to submit
        // vms and cloudlets according
        // to the specific rules of the simulated scenario
        private static DatacenterBroker createBroker() {

            DatacenterBroker broker = null;
            try {
                broker = new DatacenterBroker("Broker");
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
            return broker;
        }

        /**
```




```
	 * Prints the Cloudlet objects
	 *
	 * @param list list of Cloudlets
	 */
	private static void printCloudletList(List<Cloudlet> list) {
		int size = list.size();
		Cloudlet cloudlet;

		String indent = "    ";
		Log.printLine();
		Log.printLine("========== OUTPUT ==========");
		Log.printLine("Cloudlet ID" + indent + "STATUS" + indent + "Data center ID" + indent + "VM
ID" + indent + "Time"
				+ indent + "Start Time" + indent + "Finish Time");
		Log.printLine("========== EducaCiencia FastCode ==========");
		Log.printLine("Testes OK");

		for (Cloudlet value : list) {
			cloudlet = value;
			Log.print(indent + cloudlet.getCloudletId() + indent + indent);

			if (cloudlet.getStatus() == Cloudlet.SUCCESS) {
				Log.printLine("SUCCESS");
				Log.printLine("========== EducaCiencia FastCode ==========");
				Log.printLine("Testes OK");

				DecimalFormat dft = new DecimalFormat("###.##");
				Log.printLine(indent + indent + cloudlet.getResourceId() + indent +
indent + indent + cloudlet.getVmId()
						+ indent + indent +
dft.format(cloudlet.getActualCPUTime()) + indent + indent
						+ dft.format(cloudlet.getExecStartTime()) + indent +
indent
						+ dft.format(cloudlet.getFinishTime()));
			}
		}

	}
}
```

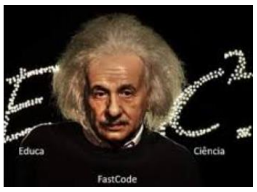
```
/* ******************************************
 * ************** output exemplo **************
 * ******************************************
Starting NetworkExample1...
Initialising...
Topology file: topology.brite
Problem in processing BRITE file. Network simulation is disabled. Error: topology.brite (O sistema não pode
encontrar o arquivo especificado)
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
160.1: Broker: Cloudlet 0 received
160.1: Broker: All Cloudlets executed. Finishing...
160.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
========== EducaCiencia FastCode ==========
Testes OK
    0        SUCCESS
========== EducaCiencia FastCode ==========
Testes OK
        2           0        160        0,1        160,1
NetworkExample1 finished!
 ******************************************/
```





Mais detalhes e artigos técnicos:

Cloudsim é uma iniciativa para a comunidade de pesquisa em computação em nuvem, com o objetivo de compartilhar o conhecimento prático sobre o Cloudsim ou suas implementações de casos de uso de simuladores de extensão (como iFogsim, Workflowsim, cloudsimSDN, etc.) e, esperançosamente, servirá como documentação do Cloudsim. Além disso, também publicamos um curso on-line autodidata intitulado “Essential Cloudsim Tutorials” que ajudará você a seguir os insights detalhados de implementação para as várias abordagens baseadas em simulação do Cloudsim básico (por exemplo) Agendamento de tarefas, agendamento de máquina virtual, gerenciamento e otimizações de recursos, otimizações de computação com consciência de energia, etc.
A seguir estão algumas das últimas postagens que foram publicadas como documentação do cloudsim de cenários relacionados:
Entidades do iFogsim – Este artigo resume várias classes de entidades disponíveis no kit de ferramentas de simulação do iFogsim e como elas estão diretamente relacionadas ao mecanismo de simulação do kit de ferramentas de simulação básico do Cloudsim.
Estrutura do Projeto iFogsim: Um Guia para Iniciantes – Este artigo servirá como um guia para entender o kit de ferramentas da API de estrutura do projeto iFogSim. É absolutamente necessário estudar os namespaces do projeto para entender que tipo de classes existem e como elas são agrupadas.
Cloudsim – O Cloudsim é um kit de ferramentas de simulação que suporta a modelagem e simulação da funcionalidade principal da nuvem. Este artigo discute a visão geral do kit de ferramentas do Cloudsim e ajuda você a começar,



juntamente com referência aos recursos relevantes e tutoriais relevantes do Cloudsim.

Configuração do CloudSim usando Eclipse – Um tutorial passo a passo do Cloudsim para instalar/configurar o Cloudsim Simulation Toolkit usando o Eclipse IDE. Este tutorial de configuração do Cloudsim começa com os pré-requisitos necessários e leva você na jornada até a execução do seu primeiro exemplo do Cloudsim.

CloudSim Simulation Toolkit: Uma introdução – Um artigo abrangente de leitura obrigatória que dará uma visão sobre os fundamentos essenciais do Cloudsim Simulation Toolkit, que é uma importante ferramenta de simulação para pesquisadores que trabalham na área de computação em nuvem.

Migração de máquina virtual no Cloudsim – Para pesquisadores baseados em nuvem, o artigo ajudará a apreciar o processo passo a passo da implementação da migração de máquina virtual feita no Cloudsim.

Criar um evento definido pelo usuário e CloudsimTags – simular um evento definido pelo usuário no Cloudsim é um requisito importante para a implementação do cenário do usuário e é crucial definir os Cloudsimtags correspondentes.

Cloudlet na simulação Cloudsim – O Cloudlet no Cloudsim define os atributos relacionados à carga de trabalho que deve ser simulada no mecanismo de simulação, por exemplo, comprimento da instrução, recurso, etc.

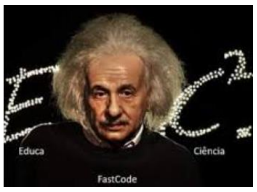
Cenário de simulação com reconhecimento de energia no Cloudsim – O suporte ao cenário de simulação com reconhecimento de energia no Cloudsim é um dos maiores avanços e tem sido amplamente utilizado pela comunidade de pesquisa para publicar seus resultados otimizados relacionados aos seus algoritmos propostos referentes a migrações de VM, SLAs, alocação de recursos, redução do consumo de energia, etc.

Guia para o fluxo de trabalho de simulação CloudsimExample1.java – Este artigo ajudará você a entender o fluxo de trabalho de simulação CloudsimExample1.java com uma discussão detalhada sobre o código de cloudsimexample1.java.

Guia para iniciantes sobre a estrutura do projeto Cloudsim – Este artigo é um guia para iniciantes para começar a usar o kit de ferramentas de simulação do Cloudsim. Antes de se aprofundar no código, é essencial entender qual namespace contém qual tipo de classes e como elas são agrupadas.

Como fazer máquina virtual e agendamento de tarefas no CloudSim – Na computação em nuvem, o agendamento é um tópico interessante. A estrutura do kit de ferramentas de simulação do CloudSim abordou esse caso de uso e forneceu um conjunto de hierarquia de classes que especifica o mecanismo básico de agendamento com relação ao timeshare e ao spaceshare. Este artigo explica sobre o mesmo e como ele pode ser estendido.

Este site será mantido por Anupinder Singh , que publicou o site www.superwits.com no ano de 2013 e contribuiu com o conteúdo introdutório básico sobre o Cloudsim Simulation Toolkit por meio de uma lista de reprodução no canal do YouTube .



Instalar o NetBeans

Baixar o arquivo JAVA: O Java Development Kit (JDK) é necessário para executar aplicações Java. Ele inclui o compilador Java e a máquina virtual Java.
Por que: O CloudSim é escrito em Java, então você precisa do JDK para poder executá-lo.

Baixar o arquivo CloudSim: O CloudSim é um framework de simulação em nuvem open-source. Ele permite modelar e simular diferentes tipos de nuvens e aplicações.
Criação do projeto no NetBeans com o nome "Redes": Um projeto no NetBeans é um contêiner para todos os arquivos relacionados a uma aplicação. Você criará um novo projeto para organizar os arquivos do seu primeiro exemplo CloudSim.

Adicionar o primeiro exemplo do CloudSim: O CloudSim vem com vários exemplos para te ajudar a começar.
Você irá copiar o código do exemplo para o seu novo projeto e adaptá-lo conforme necessário.

Configurar o JAR: Um JAR (Java Archive) é um formato de arquivo usado para agrupar vários arquivos Java em um único arquivo.
O CloudSim é distribuído como um JAR, e você precisa adicioná-lo ao seu projeto para que o NetBeans saiba onde encontrar as classes do CloudSim.

Abra o NetBeans: Execute o NetBeans.

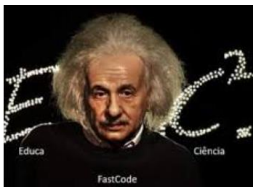
Crie um novo projeto: Vá em "File" -> "New Project". Selecione "Java" e escolha o tipo de projeto mais adequado (por exemplo, "Java Application"). Dê o nome "xxxxxx" ao seu projeto.

Importe o exemplo CloudSim:
Localize a pasta "examples" dentro da sua instalação do CloudSim.
Copie o arquivo Java do exemplo que você deseja utilizar para o seu projeto " xxxxxx ".

Adicione o JAR do CloudSim:
Clique com o botão direito no seu projeto " xxxxxx " e vá em "Properties".
Na seção "Libraries", clique em "Add JAR/Folder".
Localize o arquivo JAR do CloudSim e adicione-o.

Execute o exemplo:
Clique com o botão direito no arquivo Java do exemplo e selecione "Run File".

Observações importantes:
Versões e compatibilidade: Certifique-se de que as versões do NetBeans, Java e CloudSim são compatíveis.



Configurações específicas: Dependendo da versão do CloudSim e do exemplo escolhido, pode haver configurações adicionais necessárias. Consulte a documentação do CloudSim para mais detalhes.
Adaptação do código: O exemplo que você copiar pode precisar de algumas adaptações para atender às suas necessidades específicas.

Recursos adicionais:
Documentação do CloudSim: A documentação oficial do CloudSim contém exemplos mais complexos e explicações detalhadas sobre as diferentes funcionalidades.
Documentação oficial do CloudSim: http://www.cloudbus.org/cloudsim/