Final Report

by Yue Wang

Abstract

Cloud computing is one of the advanced and powerful computing model that satisfies enormous needs in business and science area. Due to the complicated environment factors and increasing customer requirements, resource allocation and job scheduling are considered as two highly important processes to improve efficiency.

The traditional job scheduling strategy of cloud environment needs to be optimized to suit more requirements. This paper systematically studied cloud computing technology, greedy algorithm, and cloud resource allocation. A system job schedule has been created based on greedy algorithm and greedy model structure. This algorithm aim at binding the job with highest priority to the most appropriate machine and building balance between resource allocation and job scheduling.

The project follows user's QoS preference by assigning user tasks to different groups. In each queue, the equity of resource allocation can be calculated by an evaluation formula.

All the simulation has been tested using CloudSim. We added new attributes and functions to Cloudlet. In addition, We implemented greedy algorithm of binding cloudlet to vm in DatacenterBroker Class and recompile the CloudSim source code to achieve the proposed job scheduling strategy.

Keywords: Cloud Computing, Greedy Algorithm, Resource Allocation, Job Scheduling, QoS.

Catalog

1.0 Introduction

Cloud Computing is a newly developed technology. It has high commercial value and potential. The main core technologies and features of cloud computing are resource allocation and job scheduling. This paper showed the basic theory of greedy algorithm, and the implementation of corresponding functions.

1.1Background

1.1.2 Cloud computing

Cloud computing can be described as a system platform or a type of application that dynamically allocates and arrange provision, configuration, reconfigure, and deprovision. Servers can either be physical or virtual. A complete cloud also includes other computing resources like SANs, network devices, firewalls and other security devices. According to Frantsvog[1](Frantsvog,2012), cloud computing can best be described as a giant pool, which contains all the resources that can be accessed through the 'Cloud' in any time. Compared to traditional IT, cloud computing is much faster, flexible.

1.1.3 Fog computing

Stojmenovic said [2](Stojmenovic, 2016) fog computing is a paradigm that extends Cloud computing and services to the edge of the network. Fog is similar to cloud, it provides data, compute, storage and application services to end users. In Fog computing, services are mostly performed at end sides. This new distributed computing allows applications to run as close as possible to sensed actionable and big data. In other words, Fog computing is like a Cloud computing close to the ground,creates automated response that drives the value[2].

1.1.4 Difference between Fog and Cloud

Cloud computing using massive computes and virtualize technique to accelerate the computation of huge tasks with shared physical resource. It efficiently saves the cost of computes and maintenance. But Cloud computed is limited by internet bandwidth and coverage area. Sometimes, cloud computing could not satisfy the request of real-time and high liability demands. The invent of fog computing improves this situation. Fog computing puts tasks to fog server as much as possible. By transferring data on wearable devices, fog computing reduces the total delay between server and clients, and able to provide services without internet connection.

1.1.5 SLA

A service level agreement (SLA) is a contract between a service provider (either internal or external) and the end user that defines the level of service expected from the service provider[8].LAs are output-based in that their purpose is specifically to define what the customer will receive. SLAs do not define how the service itself is provided or delivered. The SLA an Internet Service Provider (ISP) will provide its customers is a basic example of an SLA from an external service provider.

1.1.6 Related work:

From Beloglazov's work[7] (Beloglazov, 2012), his work focus on simulating energy saving strategies by using a virtualized data center. He determined a single virtual machine migration problem and use online algorithms to minimize the cost of energy consumption. His research proved the cost of the optimal offline algorithm by analyzing the range of the cost function for the domain of all possible algorithms and he also analyzed three cases when m < v. m<=v, and m >v. And he tested the dynamic virtual machine consolidation problem, and proved the ratio of the optimal online deterministic algorithm.

From Motiwani's work[8](Motiwani, 2015), he identified server methods in the migration to the new energy efficient cloud architectures. To solve the VM Consolidation problem, he observed the non-overload and overload states of the host and he used a dynamic heuristic of maximum utilization. When the process of detecting overload or underload detection is invoked, it compares the current CPU utilization with dynamic heuristic of the host , with the defined threshold. Then, he select appropriate VMs to migrate from that host.

1.2 Job scheduling in Cloud

Job scheduling and resources allocation is a well discussed problem in Grid Computing, Parallel Computing and Distributed Computing. And Cloud computing extended this problem.

Comparing Cloud Computing and Grid Computing, Cloud Computing is like a huge centralized datacenter and Grid Computing is like a scattered datacenter. Cloud Computing has more commercial use and various services while Grid Computing only focus on science [3](Foster,2008).

Cloud Computing has many new features compared to traditional computing. It also has been concerned by many problems. And job scheduling and resources allocation is the mainly concerned problem. Many algorithms are used to solve this problem. Generally there are three kinds of algorithm, they are based on efficiency, fairness and economy. The one based on efficiency focus on completing jobs in the minimum time. The one based on fairness focus on client's satisfaction. And the one based on economy is the most complicated one.

1.3 QoS

Based on commercial use of cloud computing, quality of service is an important factor in task scheduling.

QoS(Quality of Service) is a factor that describes the performance of network service [9](Xu, 2009). It is very important referring to customer satisfaction. Unlike traditional criteria(cost performance, efficiency), the service of cloud computing is measured by quality and satisfaction of various user requirements. It provides multiple services to internet users. According to different preferences, cloud computing provides classification of resources precisely. For example, some users prefer high efficiency while others may require comparatively low prices. Following the requirement and specification of QoS, users are classified into different groups. Thus, users could get their desired services and resources[10](Calheiros,2011),in terms of bandwidth, expectation finish time, reliability, and cost.

1.4 Algorithm based on efficiency

1.4.1 Greedy algorithm

The goal of greedy algorithm is to provide partially optimised solution at certain circumstance. In other words, it provides currently best solution. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.[4]

1.4.2 Genetic algorithm

Genetic Algorithm is derived from the theorem of biological evolutionism that invented by Holland in 1978. This method repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution[11](Gu,2012).

By the optimization method of probability, it can automatically obtain and instruct the optimized searching space and adjust the searching direction by itself. Considering the VM resources scheduling in cloud computing environment and with the advantage of genetic algorithm, this algorithm presents a balanced scheduling strategy of VM resources based on genetic algorithm. According to historical data and current states, this method computes in advance the influence it will have when the current VM service resources that need deploying are arranged to every physic node, based on which the method achieves the best load balancing.

1.4.2 Ant colony optimization algorithm

The ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.

This algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations.

From li's work[12](Li,2011),they have proposed the ant colony optimization algorithm for achieving tasks scheduling with load balancing, and they have experimentally evaluated the ant colony optimization algorithm in applications with the number of tasks varying from 100 to 500. The experimental result shows that the ant colony optimization balance the entire system load effectively. Weather the sizes of the tasks are the same or not, ant colony optimization can handle all conditions, and outperforms FCFS and ACO algorithms in cloud computing environment.

1.5 Main Idea

The main idea comes from Choudhary's study[5](Choudhary,2012). In his research, he performed a dynamic optimization for task scheduling in cloud environment. He considered priority and cost of task. For each prioritized task, he calculated the turnaround time by taking three parameters, waiting time, task length and processing power of virtual machine. Then the virtual machine with minimum turnaround time that is capable to execute the task is selected and task is scheduled for execution on that machine.

Enlightened from his work, the idea of using greedy algorithm to schedule tasks has showed in my mind. The basic thought is to add priorities to each cloudlet. Then resolve the problem by getting the time cost of each cloudlet that runs on different machines. After that, we calculate the best solution for the current task. And finally, combine them to get an ultimate solution.

1.6 Data sets

Cloudlet ID	length	File size	output size	Priority
0	**900**	**300**	**300**	**9**
1	**1200**	**1000**	**500**	**4**
2	**1500**	**1000**	**500**	**4**
3	**1800**	**800**	**300**	**1**
4	**2100**	**1500**	**500**	**1**
5	**2400**	**1500**	**500**	**4**
6	**2700**	**2000**	**400**	**10**
7	**3000**	**2500**	**500**	**10**

Vm ID	MIPS	Pe	Bw
0	1000	4	2000
1	500	2	3000
2	300	2	1200
3	300	1	2000

1.7 Risk management

ID	Title	Description	Probability	Actions
01	lost of data	Accidently lost core research data or code	2	Backup cloudsim onto online drive
02	unfixable change of base code	Adding new attributes to the base code and causing unidentified error	7	Upload current work regularly onto github
03	work overdue	Can not finish work in a limited time	5	Apply for special consideration

1.8 Resource requirement

Hardware:

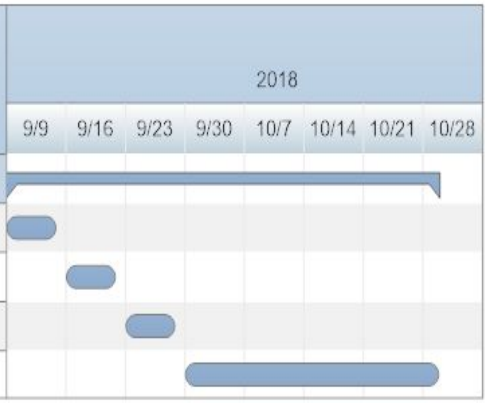
- Operating system: Windows 7/8.1/10(64-bit versions)
- Cpu: Intel Core i3-2400 or better
- RAM: 4 GB or better

Software

- Intellij
- cloudsim toolkit
- java JDK 1.6 or higher

1.7 Timeline

	Task		Start	End	Dur	
	Main Topic		9/9/18	10/29/18	36	
1	add priorities		9/9/18	9/14/18	5	
2	implement migration algorithms		9/16/18	9/21/18	5	
3	implement detecting algorithms		9/23/18	9/28/18	5	
4	test and write report		9/30/18	10/29/18	21	



2.0 Methods

To keep the simulation as simple as possible, this project only take mips and cloudlet length as core parameters. The result is easy to observe and understand.

2.1 Expected finish time

Since we only consider MIPS in this case, users expect that they should get the averagely equaled resource as others. So the expected finish time should be the length of task divides average MIPS of all Vms.

$$Eft = \text{cloudlet length/average MIPS}$$

2.2 J factor

Generally, Cloud providers provides network services to customers and the cost depends on usage. From customer's perspective, they require to have reasonable resources and efficiency. We can use J function to calculate the fairness of resource allocation.

$$J = In(AR/ER)$$

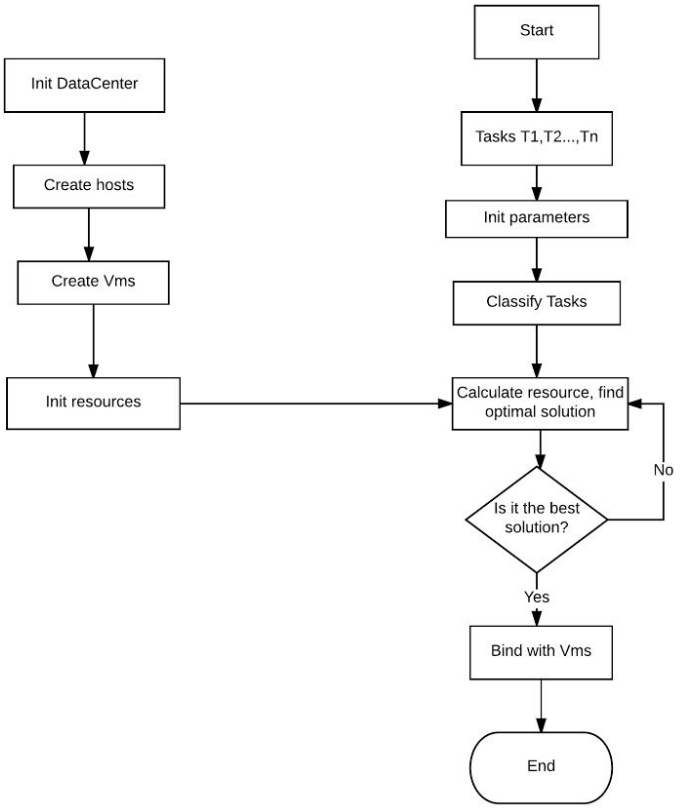
The concept is simple, AR(Actual Resource) presents the actual resource that has been allocated to clients. ER(Expected Resource) presents the expected resource that user want.

case: AR = ER, j = 0 It is the most fair situation, which means that the actual allocated resource is as same as user expected.

case:AR > ER, j > 0 It shows that the machine takes more time to finish the task than user expected. It is unfair to users.

case:AR < ER, j < 0 In this case, user got more resources than expected. It provides better services to specific users and sacrifices other user's benefits.

2.3 Class Diagrams



2.3.1 Sorting

This project uses CloudSim as the simulation tool. We added variables like actualTime, expectationTime. To sort cloudletList and vmList, we imported build-in modules, Collections and Comparator. The code shows below.

```
Sort cloudletList in priority ascending order
Sort vmList in MIPS descending order
*/
Collections.sort(cloudletList, new Comparator<Cloudlet>() {
  @Override
  public int compare(Cloudlet o1, Cloudlet o2) {
    return o2.getPriority()-o1.getPriority();
  }
});
Collections.reverse(cloudletList);


Collections.sort(vmList, new Comparator<Vm>() {
  @Override
  public int compare(Vm o1, Vm o2) {
    return Double.compare(o1.getMips(),o2.getMips());
  }
});
```

2.3.2 Algorithms

Sort cloudletList in priority ascending order
Sort vmList in Mips descending order

```
//calculate time cost of each cloudlet spend on each vm
for(i = 0, i < cloudletList length, i ++){
        for (j = 0, j < vm number, j ++){
                time[i][j] = cloudlet length[i]/ Vm MIPS[j]
        }
}

//Allocate the cloudlet with highest priority to the fastest Vm
for(i = 0, i < cloudlet num, i++){
        set estimated minimum load
        for(j = 0 , j < vm number, j ++){
                compare minload with currentload and get optimal solution
        }
        bind cloudlet to vm
}
```

2.3.3 Simulation

//init CloudSim

```
CloudSim.init(num_user, calendar, trace_flag);
```

//Create datacenter

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
```

//create broker

```
DatacenterBroker broker   = createBroker();
```

//create vm list

```
vmList = new ArrayList<Vm>();
 //set  parameters
//create vms
//submit vmlist to broker
broker.submitVmList(vmList);
```

//create cloud list

```
cloudletList = new ArrayList<Cloudlet>();
//set cloudlet parameters
//create cloudlets
//submit clouldletlist to broker
broker.submitCloudletList(cloudletList);
```

//use greedy algorithm to bind cloudlets to vms

```
broker.bindCloudletsToVmsGreedy()
```

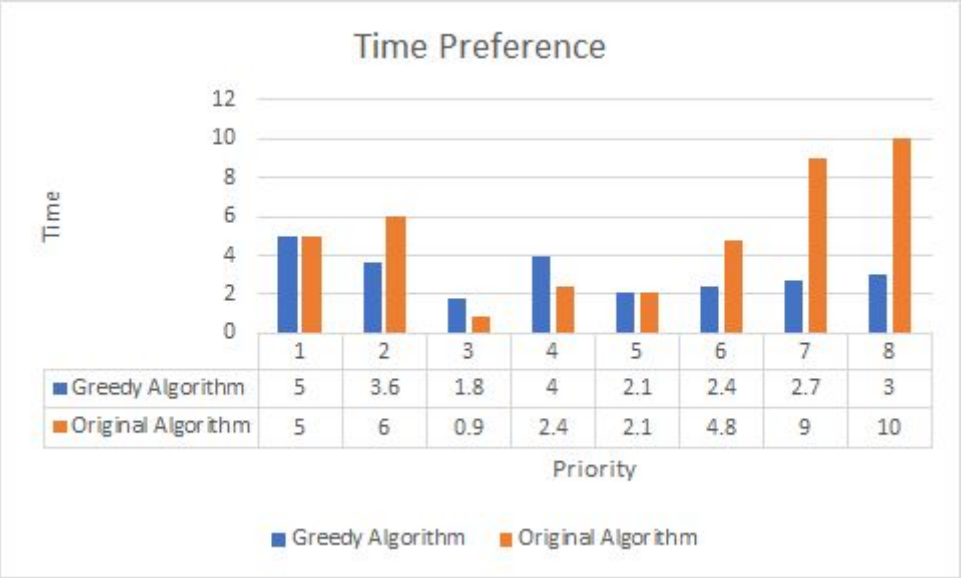
//start simulation

```
CloudSim.startSimulation();
```

//end simulation and get result

```
List<Cloudlet> newList = broker.getCloudletReceivedList();
CloudSim.stopSimulation();
printCloudletList(newList);
```

3.0 Result and Analysis



From the bar chart, we can see clearly that greedy algorithm has significant advantage when dealing with high priority tasks and it also keeps doing efficient job with low priority tasks. The performance shows that using greedy algorithm is almost three times faster than the original algorithm at priority level 7,8. When it comes at lower priority level, the time cost of our algorithm is approximately equal to the original algorithm.

Generally, the build in algorithm takes longer to complete a task in most cases.From the efficiency perspective, greedy algorithm is stable and highly efficient.

According to the J function bar chart, at most time, the J value of build in algorithm is greater than 0, which indicates that allocated resources did not reach client's expectation. It allocated nearly same amount of resources to low priority jobs and high priority jobs. However, it is a waste to spend too much resource on low priority jobs. And it obviously goes against customer's will. Conversely, the J value of greedy algorithm has done a better job. It allocated a lot more resources to high priority jobs. The J value reaches -0.5 from priority level 5 to 8 while the original algorithm paid much less resources on these tasks.

However, our algorithm spent a even larger amount of resources on some tasks with low priorities, 1 and 4. It surprisingly violated our expectation. The cause is stay unknown, and it may need to take larger data sets to test. Besides, it performed normally as expected at priority level 2 and 3.

3.1 Conclusion

By simulating using Cloudsim, our algorithm showed its power compared to the build in algorithm. It highly improved resource allocation when we considered priorities. Tasks with high priorities will have more resources and will be done firstly. It also keeps fairness to other customers that all other work has been done with similar time cost as previous algorithm. Greedy algorithm has better computational practice at analyzing resources and task scheduling. It also satisfies customer expectations.

4.0 Conclusion and future work

4.1 Conclusion

This paper studied greedy algorithm and compared it with cloudsim original resource allocation algorithm. From the comparison, our algorithm is obviously more efficient than the traditional algorithm. It also gives customer a better experience and convince them with the fairness resource allocation.

we considered two major factors for resource allocation and job scheduling, efficiency and user satisfaction. J factor has been used to measure the difference between those two factors.

CloudSim has been used to simulate the test of our algorithm, we extend cloudSim by add new variables to cloudlet and functions to get expectation time. We also reload the function “bindCloudletToVm()” in DatacenterBroker class.

4.2 Future work

This project is based on user's perspective. Our greedy algorithm module only focused on time efficiency and resource fairness. To calculate those features, we only concerned task priorities and virtual machine's MIPS.

In future, the project could do deeper research on more features. And J factors could be affected by many other factors such as virtual machine's bandwidth, number of Pes, ram. To get a more completed and accurate result, we could also consider cloudlet file size, output size, as well as host MIPS and CPUs.

Based on geolocation and green economy, it is possible to add more attributes than just priority.Such as, adding distance between virtual machines and tasks, calculating CPU cost, breaking down each step of algorithm to real world factors.

References:

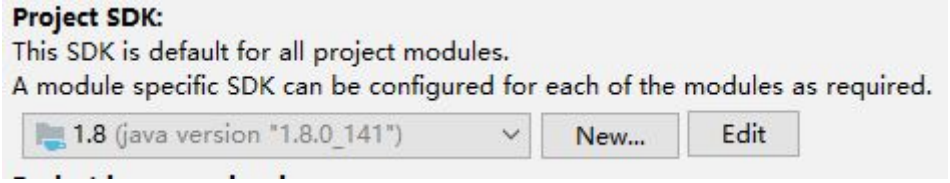
[1]Frantsvog, Dean ; Seymour, Tom ; John, Freneymon
International Journal of Management & Information Systems (Online), 2012, Vol.16(4), p.317
[2]Stojmenovic, Ivan ; Wen, Sheng ; Huang, Xinyi ; Luan, Hao
Concurrency and Computation: Practice and Experience, July 2016, Vol.28(10), pp.2991-3005
[3]Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008, November). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08* (pp. 1-10). Ieee.
[4]https://en.wikipedia.org/wiki/Greedy_algorithm
[5]Choudhary, M., & Peddoju, S. K. (2012). A dynamic optimization algorithm for task scheduling in cloud environment. *International Journal of Engineering Research and Applications (IJERA)*, *2*(3), 2564-2568
[6]https://www.paloaltonetworks.com/cyberpedia/what-is-a-service-level-agreement-sla
[7]Beloglazov, A., & Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, *24*(13), 1397-1420.]
[8]Motwani, A., Patel, V., & Patil, V. M. (2015). Power and QoS Aware Virtual Machine Consolidation in Green Cloud Data Center. *International Journal of Electrical, Electronics and Computer Engineering*, *4*(1), 93.
[9]Xu, M., Cui, L., Wang, H., & Bi, Y. (2009, August). A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. In *Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on* (pp. 629-634). IEEE.
[10]Calheiros, R. N., Ranjan, R., & Buyya, R. (2011, September). Virtual machine provisioning based on analytical performance and QoS in cloud computing environments. In *Parallel processing (ICPP), 2011 international conference on* (pp. 295-304). IEEE.
[11]Gu, J., Hu, J., Zhao, T., & Sun, G. (2012). A new resource scheduling strategy based on genetic algorithm in cloud computing environment. *Journal of Computers*, *7*(1), 42-52.
[12]Li, K., Xu, G., Zhao, G., Dong, Y., & Wang, D. (2011, August). Cloud task scheduling based on load balancing ant colony optimization. In *Chinagrid Conference (ChinaGrid), 2011 Sixth Annual* (pp. 3-9). IEEE.

Instructions

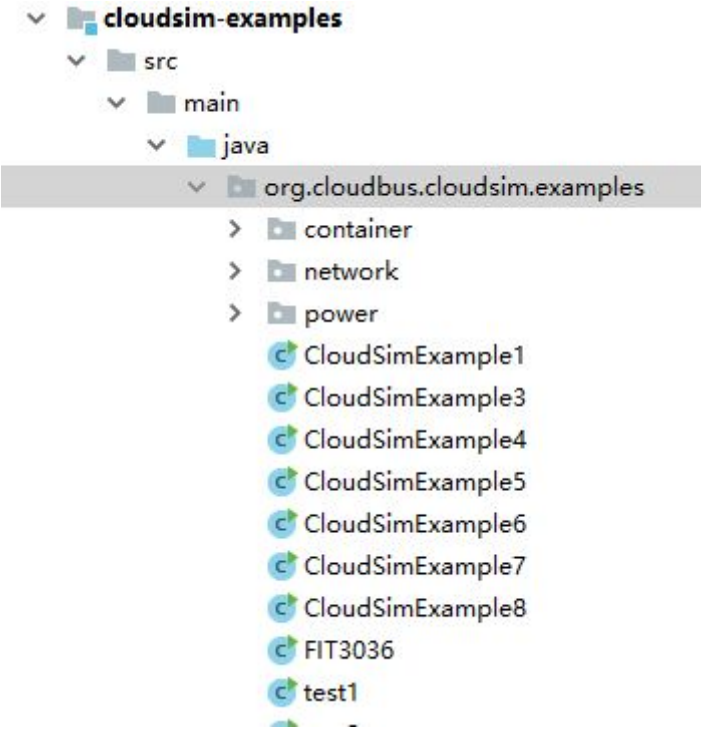
1.0 Installation

1.1Unpack zip file(cloudsim-cloudsim-4.0)

1.2Set SDK



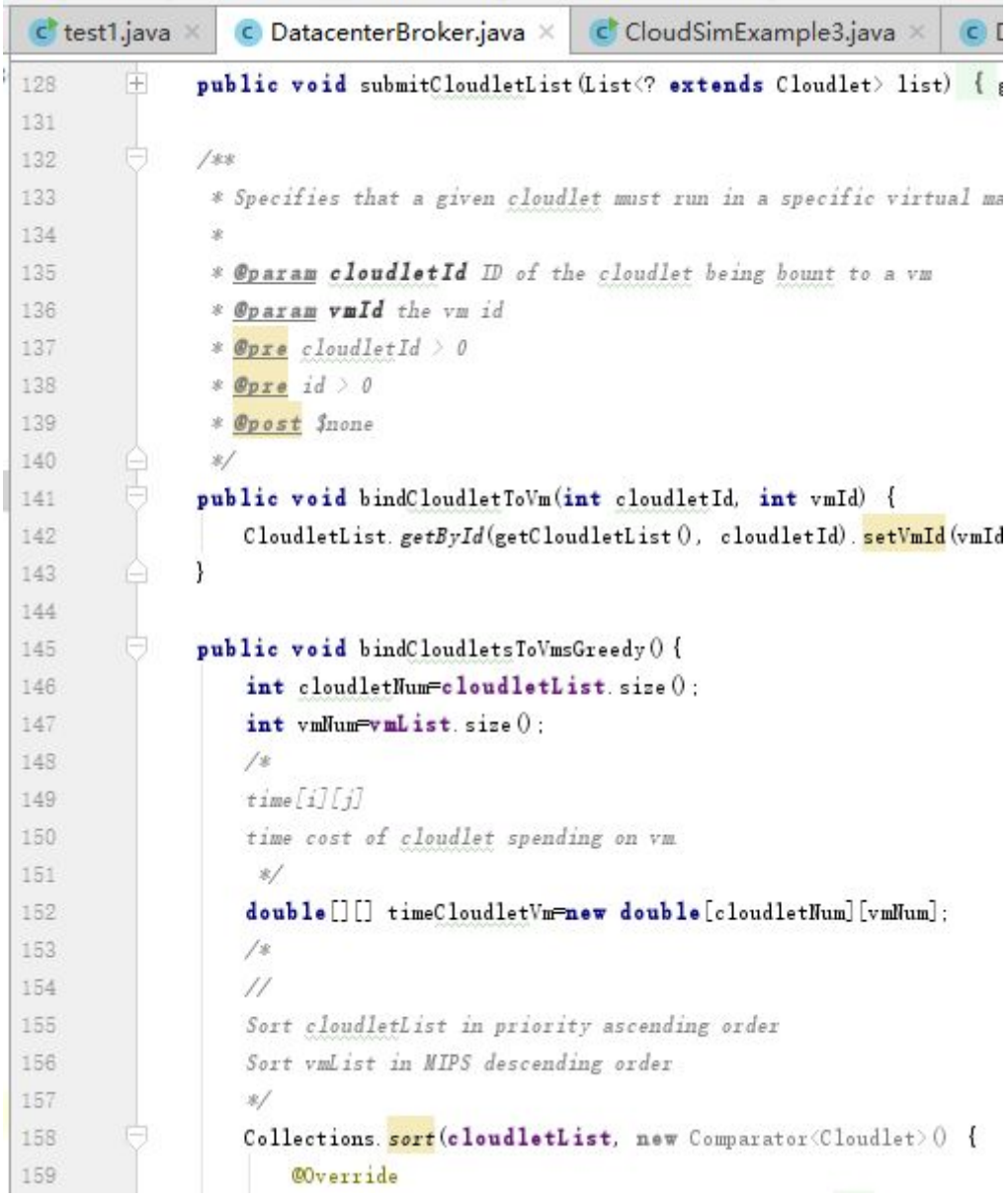
2.0 Go to modules - cloudsim-examples- srs- main -java - test1



3.0 Run test1

4.0 Core algorithm

Our core algorithm is in DatacenterBroker class, line 145



The function to get J value is in Test1 Class, line 231