VU E-Commerce

Assignment 188.427 VU E-Commerce 2016WS

Filip Rydzi BSc

Institut fuer Softwaretechnik und Interaktive Systeme

1 Introduction

The incoming era of Internet of Things (IoT) brings bilions of new heterogenous devices (things) connected to the internet [2]. The apps developed for these devices (things) may have critical requirements on low-latency, high bandwith E.g. an autonomus car can generate large amount of data (1 Gigabyte) every second and need to process them to make a correct decision [7]. This is a compute intensive operation requiring to be processed in a very short time, thus cloud computing is not sufficient for this task, because it would take a long response time. Therefore the concept of edge computing has been developed. The idea of edge computing is to offload such compute intensive operations to the edge of the network, near to the user, s.t. they can be processed (and return response) in a short time. An edge is a small datacenter located in a short physical distance to the users (at the buildings in urban areas...), so the latency for the data transfers is rather low - usually the data gets processed near the location where it was originally generated. This brings many benefits for the deployed app, as: low-latency, high-bandwidth, privacy (data dont need to travel such a long distance, therefore arent exposed to many possible attack locations)... [8] Because an edge is a simple data center (without UPS systems), which could potentially fail, there is a need to develop a strategy for handling these failures, without disrupting a running application.

Your task in this assignment is to develop an edge controller to efficiently distribute the workload among the edges by considering:

- expected workloads
- failures, which are to occur
- load balancing and migration of VMs

2 Infrastructure

2.1 Resources(edge computing infrastructure)

VMs

- Size, consumed memory, consumed CPU, consumed network bandwidth (depends on the consumed memory), page dirtying rate (depends linearly on the combination of the utilized memory, CPU and network bandwidth)

– Running time given by a normal distribution function
– Origin of the request
  • This is the location of the user, who is served by this VM

PMs

– Size, consumed memory, consumed CPU, consumed network bandwidth.
– Energy signature:
  • linear combination of the CPU, memory and network workload, but not equal to zero in the idle state. It depends on the consumed resources, e.g. the same workload of CPU and network will end up into different energy consumption rates on different machines.Energy utilization function of a machine is given as follows:

$$U = U_0 + W_{cpu}U_{cpu} + W_{mem}U_{mem} + W_{network}U_{network} \quad (1)$$

where U is the total energy utilization, U_0 is the energy utilization in the idle state when no resources are consumed, U_i is the energy utilizations considering resources at the highest possible workload, W_i are workload rates.

Micro-datacenter(edge)

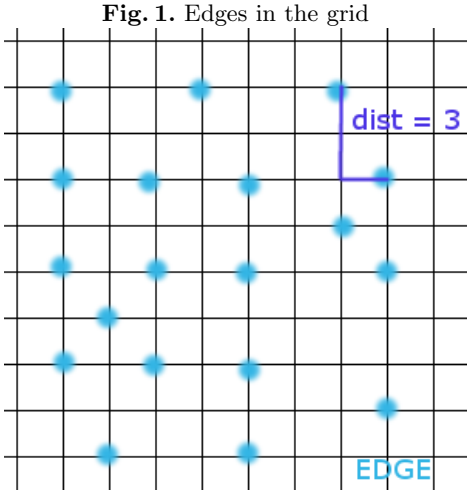
– Includes max 10 PMs, but number of running PMs differs due to the workload.
– Location of the edge is known.
– The edges are placed in a grid as shown in Figure 1. At most one edge can be placed at a position in the grid.
– The distance between each two edges is defined as Manhattan distance between the edges.
– The edges are connected between each other. Network bandwidth is defined for each connection between two edges.

$$U = U_0 + \sum_{m \in PM} U_m \quad (2)$$

– U is total energy utilization, U_0 is the energy utilization when no PMs are running, U_m is energy utilization of running PM m.

Big-datacenter (optional) This component can serve as a failure detector at the PMs/edges and performs actions defined by your strategy to recover from failure.

Fig. 1. Edges in the grid



2.2 SLAs

SLAs are important as a contract agreed between service provider (you in this assignment) and the customer to define particular aspects of the service (non-functional requirements, qualities...). All functionalities specified in the functional requirements should be delivered with respect to these qualities and non-functional requirements (e.g. an operation can take at most 5 seconds...).
Although the SLAs are strictly defined, sometimes you may be speculative and you can break the SLAs to save your costs. This makes advantage if e.g. your customer doesn't utilize all resources all the time and for you it may be cheaper to pay him a penalty for breaking the SLAs, rather then hosting so many (non-utilized) resources all the time.
In this assignment you should select the SLAs for your infrastructure and based on the simulation evaluate how your algorithm can handle the failures compared to the baseline under these SLAs (e.g. at availability 90% your algorithm is able to handle 1000 failures, but the baseline only 800...) see section 3 Tasks.
Sample SLAs follow:

- Agreed VM characteristics: CPU, memory, network bandwidth (e.g. CPU: 3GHz, memory: 4GB, network bandwidth: 54 Mbit/s)
- Network bandwidth between the edges (e.g. 600 Mbit/s)
- Mean time to recover from failure [9]

$$MTTR = \frac{totalDownTimeCausedByFailure}{numberOfBreakdowns} \tag{3}$$

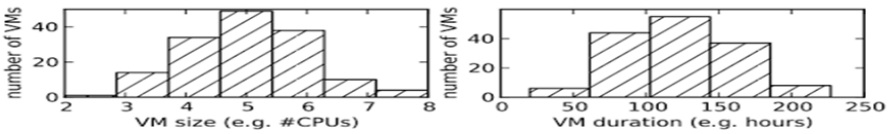
- Latency (e.g. 70ms)
- Availability of your infrastructure (e.g. $> 99.95\%$)

$$availability = \frac{uptime}{uptime + downtime} \tag{4}$$

2.3 Workload

We assume that requests for VMs were generated randomly by uniformly distributing the creation time and duration. The specifications of the requested VMs were modelled by normally distributing each resource type (CPU, memory,). An example VM request distribution is illustrated in Figure 2.

Fig. 2. VM request resource and duration histogram



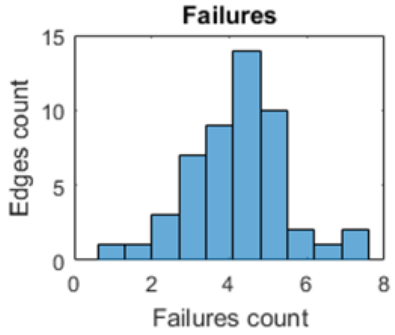
2.4 Failures

We identify three types of failures in cloud platform: hardware/PM failure, VM failure, application faiure [6]. For this assignment we extend these types by adding edge failure (all PMs running there fail). We concentrate on the hardware/PM and edge failures. In a case of PM failure the VMs running in the failed PM, should be handled according to your developed strategy and PM needs to be restared for recovery. You also need to consider the case when the whole edge fails (all PMs running here need to be restarted).
You may assume there is an anomaly detection system, which predict an occurrence of a failure before it really happens, such that you can perform the steps necessary to recover from failure.
Figure 3 shows a distribution of the failures accross the edges: the average number of failures occur in the most of the PMs/edges, the min/max number of failures occur in a small amount of PMs/edges.

2.5 Migrations

For the edge computing its critical to perform the migration of a VM in an effective manner. Its very important that the VM is being migrated as the user moves, s.t. the running app is still in a small physical distance to the user, and the existing connection shouldn't be disrupted. Live VM migration technologies could be a good candidate for these requirements. As a baseline for VM migrations in this assignment, you can use the simplified version of pre-copying algorithm for live VM migrations: In the first step, this approach creates an image of VMs physical memory and pushes it accross the network to the destination. Since VM is still running while sending the image, some pages of the memory are being dirtied, these dirtied pages must be re-sent to destination. During the second sent operation some pages are being dirtied again and must

Fig. 3. Failures in edges histogram



be re-sent to the destination. This process is repeated until the amount of dirtied pages falls below a defined threshold. Then in the last step, the VM is stopped for a short time and pages dirtied in the previous round are sent [5].

Formal model

- V_i - Data volume transmitted at round i
- T_i - Time consumed to transfer volume at round i
- D - Memory dirtying rate during migration (you defined this in your VM, see VMs in 2.1)
- R - Memory transmission rate during migration

Choose a memory transmission rate as a function of distance between source and destination hosts: R = func(dist) (less distance, higher memory transmission rate).

Migration process

- if $i = 0$ (the first step: pushing physical memory image to destination)

$$V_i = V_{mem} \tag{5}$$

$$T_i = \frac{V_{mem}}{R} \tag{6}$$

- if $i \neq 0$ (next steps: dirtied pages are being re-sent)

$$V_i = D \cdot T_{i-1} \tag{7}$$

$$T_i = \frac{D \cdot T_{i-1}}{R} \tag{8}$$

Simplified energy model

$$V_{mig} = \sum_{i=0}^{n} V_i \quad \text{total data volume transferred during migration} \tag{9}$$

$$U_{mig} \approx V_{mig} \quad \text{denotes the energy consumed during migration} \tag{10}$$

Goal: Minimize the U_{mig}.
In real world the U_{mig} is a linear function in form of $U_{mig} = \alpha V_{mig} + \beta$, where parameters α, β can be trained using linear regression [5].
There are other better approaches used for VM migration in edge computing e.g. VM handoff, which basically resembles the VM live migrations but uses pipelines to speed up the process of migration [3]. Although it differs in some significant aspects, for the sake of simplicity we are using the live migrations as a baseline for this assignment.

3 Tasks

Presentation 1: 19.10.2016

- Find your peers (groups of 3 people)
- Create a git repo (add eclecture-tutor@ec.tuwien.ac.at as guest/viewer)
- Select a baseline for your algorithm to handle the failures (you can use one of these strategies [4] as the baseline)
- Give some ideas for improving the baseline, which you will use later for your algorithm

Presentation 2: 16.11.2016

- Develop an algorithm to handle the failures, which occur in a random manner as described in section 2.4, your algorithm should distribute the workload from the failed PM/edge to other running PMs/edges without experiencing any notable performance degradation.
  - Consider a case of cascading failures i.e. the PM you have chosen as a destination for the workload from the failed PM/edge has also failed in a meantime
  - You should maximize the number of failures your infrastructure is able to handle (under certain SLAs) utilizing your algorithm, minimize the latency and energy consumption (as described in section 4 Key metrics)
  - For effectively distribution of a workload considering the consumed energy you can use the BFD heuristics as a baseline [1]
- Present and justify your assumption, whether you should get better/worse results utilizing your algorithm against the baseline.
- Present a methodology how to test your algorithm.
- Select a sampling time of your algorithm and justify the selection (e.g. every minute, every 5 minutes, every hour...).

- Develop a controller, which is able to handle the failures utilizing
  - your algorithm
  - the baseline
- Present an infrastructure you developed to perform a simulation.
  - HINT: you should model the edge computing infrastructure in this step as described in the 2.1 Resources section.
- Discuss and justify your implementation choices like programming language, etc.
- Select SLAs.

Presentation 3: 14.12.2016

- Perform first simulation for a short period of time (a couple of hours), this simulation should
  - generate large amount of workload as described in section 2.3 Workload
  - generate the failures as described in section 2.4 Failures
  - randomly (utilizing uniform distribution) distribute the workload to the running edges, you can assume that workload is distributed effectively, i.e. each VM is running near the end-user
  - handle failures occurred during the simulation
  - (run this simulation once utilizing your algorithm and once the baseline)
- Based on the results of your first simulation show if your assumption (from step 2 in Presentation 2) was correct or not and explain why.
  - Based on the results, give some suggestions for improvements.

Presentation 4: 18.01.2017

- Perform second simulation for a long period of time (at least one week), this simulation should
  - generate large amount of workload as described in section 2.3 Workload
  - generate the failures as described in section 2.4 Failures
  - use at least 3000 VMs, 1000 PMs running in 100 edges
  - randomly (utilizing uniform distribution) distribute the workload to the running edges, you can assume that workload is distributed effectively, i.e. each VM is running near the end-user
  - handle failures occurred during the simulation
  - (run this simulation once utilizing your algorithm and once the baseline)
- Based on the results of your second simulation show, what you have improved/made worse since the first simulation and explain why
  - Estimate the maximum amount of failures your controller is able to handle (under the SLAs you have selected) and compare it against the baseline.
- Present a final evaluation of your algorithm

4 Key metrics

You should focus on the following key metrics during the simulation and development of your algorithm:

- Maximize the number of failures your infrastructure is able to handle under certain SLAs (see section 2.2) utilizing your algorithm.
- Minimize latency.
  - For this assignment: latency is a function of distance between the user, who is served by a VM, and an edge, which hosts this VM (smaller distance, lower latency). One unit of the Manhattan distance represents 30ms of latency.
- Minimize consumed energy.

References

1. Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.
2. Dave Evans. The internet of things how the next evolution of the internet is changing everything. 2011.
3. Kiryong Ha, Yoshihisa Abe, Zhuo Chen, Wenlu Hu, Brandon Amos, Padmanabhan Pillai, and Mahadev Satyanarayanan. Adaptive vm handoff across cloudlets. Technical report, Technical Report CMU-CS-15-113, CMU School of Computer Science, 2015.
4. Seyyed Mansur Hosseini and Mostafa Ghobaei Arani. Fault-tolerance techniques in cloud storage: A survey. *International Journal of Database Theory and Application*, 8(4):183–190, 2015.
5. Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. *Cluster computing*, 16(2):249–264, 2013.
6. Alain Tchana, Laurent Broto, and Daniel Hagimont. Fault tolerant approaches in cloud computing infrastructures. In *Proceedings of the 8th International Conference on Autonomic and Autonomous Systems (ICAS12)*, pages 42–48, 2012.
7. Mark van Rijmenam. Self-driving cars will create 2 petabytes of data, what are the big data opportunities for the car industry? 2016.
8. Quan Zhang Youhuizi Li Weisong Shi, Jie Cao and Lanyu Xu. Edge computing: Vision and challenges. 2016.
9. Wikipedia. Mean time to recovery.