5 Experimental Evaluation

This chapter discusses experiments to evaluate M-CaMid. The main goal is to assess the M-CaMid in managing multi-cloud distributed applications by exploring elasticity at different levels of granularity and scopes. A systematic approach for performance evaluation was adopted (Jain, 1991) to accomplish the experiments. First, some steps are defined to conduct the experiments: goals and system boundaries, scenarios, services and outcomes, metrics, parameters and factors, experiment design. Then, the results are presented and discussed.

5.1 Goals and System Boundaries

The experiments' main goal is to estimate the impact of using M-CaMid over the performance of multi-cloud distributed applications. The experiment's system consists of a multi-cloud environment. Its components are a multi-cloud distributed application, a multi-cloud infrastructure composed of a set of private clouds and M-CaMid. The multi-cloud application follows a client-server architecture, where the client-side can invoke a service implemented by remote objects on the server-side. The service executes the calculation of the Fibonacci sequence. Remote objects can be hosted in a single node, in many nodes in a single cloud, or many clouds.

Since M-CaMid is a client-centric solution, some environment components do not belong to the system under evaluation: tools for IaaS management, low-level communication channel and the Internet. Public clouds are not used in the experiments because their IaaS infrastructure resources are out of the system boundaries.

5.1.1 Scenarios' Description

M-CaMid runs on scenarios where a multi-cloud application executes in a limited-growth infrastructure and is submitted to unpredictable peaks of workloads.

To face this challenge, M-CaMid supports the high availability and scalability of multi-cloud applications. The implemented distributed application follows the client/server architecture, in which a wide range of client applications (client-side) invoke remote objects' methods (server-side). The client-side is deployed on top of a VM outside of cloud domains. The remote object on the server-side delivers as a service the calculation of Fibonacci sequence. Initially, instances of *M-CaMid Naming Service* and *Cloud M-CaMid* are deployed in all cloud domains. *Cloud M-CaMid* instances register themselves in the respective *M-CaMid Naming Service*. Then, a *Node M-CaMid* instance is deployed on a VM in the respective cloud domain. It is worth noting that *Node M-CaMid* deployment can occur in more than one cloud domain. *Node M-CaMid* registers itself and its remote objects in the respective *M-CaMid Naming Service*. After registration, a remote object can be reachable by client-side instances.

The multi-cloud environment is composed of two cloud domains where remote objects are deployed. Both cloud domains have the same hardware configuration but different cloud management technologies: OpenNebula (Moreno-Vozmediano et al., 2012) and OpenStack (RackSpace, 2010). In both cases, the cloud infrastructure comprises a cloud frontend and a host server. The cloud frontend is in charge of managing hardware resources and works as an interface for cloud users, and the host server shares its resource for VMs. The OpenNebula and OpenStack frontend instances run on machines core i5 2.44GHz with 4 GB of memory, and their respective host servers are octa-core Xeon 2.93 GHz machines with 20 GB of memory.

VMs for hosting M-CaMid instances (Cloud and Node M-CaMid) are configured with one CPU core and 1 GB of memory. Initially, two VMs were deployed to execute the experiments: a server-side application node (Node M-CaMid) and a Cloud M-CaMid VM. As the experiment progresses, Cloud M-CaMid may deploy new VMs.

5.2 M-CaMid's Services and Outcomes

M-CaMid provides a set of services for managing communication and distribution of multi-cloud applications. M-CaMid must keep the multi-cloud application working properly, assuring its properties. At the same time, it manages the underlying infrastructure, providing efficient use of its resources.

It is expected that M-CaMid supports the remote object invocation assuring its correct completion within an acceptable period (response time). Otherwise, the invocation can fail or take a time higher than the pre-established threshold. The undesirable requests must be detected by M-CaMid that acts over the environment through the elasticity. Elasticity management comprises monitoring and controlling resources through replication, migration and load balancing in different domains. Undesirable results are short reaction time by M-CaMid facing unwanted events (e.g., overload) and efficient workload distribution among application's nodes. The desirable results are low overhead and error rate in remote invocation when M-CaMid's services are activated.

5.3 Experiments

Three experiments evaluate the M-CaMid operations for managing multi-cloud distributed applications:

1. *Management reaction time*: evaluates the automatic control of M-CaMid on dealing with undesired environment's behaviour. It observes how fast M-CaMid steps in the environment to keep it stable. The experiment evaluates M-CaMid's performance measuring the reaction time and response time with M-CaMid management turned off and turned on. Both scenarios are analysed by comparing each other. In the scenario turned on, the reaction time is measured in two elasticity levels (VM and application level).

2. *Elasticity resource management*: assesses the rational usage of infrastructure (virtual machines) considering coarse- and fine-grained management. It evaluates M-CaMid management by measuring the resource usage of VMs (CPU usage), observing resource over- and under-provisioning aspects. The experiment scenario takes place at two management levels: coarse- and fine-grained management.

3. *M-CaMid overhead*: assesses the overhead introduced by M-CaMid's node monitor on the application performance. The overhead is measured in two scenarios: node monitor turned off, and node monitor turned on. This experiment is limited to the monitoring system because a growing workload stresses M-CaMid. In this case, if the M-CaMid controller is on, the manager triggers the elasticity to fix the overload problem, affecting the experiment results.

5.4 Reaction Time

The experiment's primary goal is to assess how fast M-CaMid automatic management intervenes in the environment to reestablish a multi-cloud application's regular operation. It intends to observe the time spent by M-CaMid's strategies to reconfigure the environment under undesired behaviours. Metric reaction time is measured in milliseconds data units. Reaction time is measured when M-CaMid management is configured to adopt different levels (coarse- and fine-grained control) in the single cloud and multi-cloud domains. The experiment analyses differences among strategies in both domains. Fine-grained management is expected to take less time to reconfigure the environment regardless of the domain—furthermore, the experiment analyses differences in the same management strategy in both domains.

5.4.1 Hypotheses

In the experiment null hypotheses ($H0_{1..6}$), there is no difference in the reaction time (RT) when cross management strategy assumes the configurations turned off (off), coarse-grained (cg) or fine-grained (fg) levels, whether at a single cloud (sc) or multi-cloud (mc) domains.

$H0_1 : RT_{off\_sc} \cong RT_{cg\_sc}$

$H0_2 : RT_{cg\_sc} \cong RT_{fg\_sc}$

$H0_3 : RT_{cg\_sc} \cong RT_{fg\_mc}$

$H0_4 : RT_{cg\_mc} \cong RT_{fg\_mc}$

$H0_5 : RT_{cg\_sc} \cong RT_{cg\_mc}$

$H0_6 : RT_{fg\_sc} \cong RT_{fg\_mc}$

On the other hand, the alternatives hypotheses ($H1_{1..6}$) are:

$H1_1 : RT_{off\_mc} < RT_{cg\_sc}$

$H1_2 : RT_{cg\_sc} < RT_{fg\_sc}$

$H1_3 : RT_{cg\_sc} < RT_{fg\_mc}$

$H1_4 : RT_{cg\_mc} < RT_{fg\_mc}$

$H0_5 : RT_{cg\_sc} < RT\,cg\_mc$

$H0_6 : RT_{fg\_sc} < RT\,fg\_mc$

5.4.2 Metrics, Parameters and Factors

The metric adopted to assess the mean reaction time (RT) is the time taken by M-CaMid to conclude management actions to restore the multi-cloud application's regular operation. This period comprises the undesired event detection (application's response time violation) until the multi-cloud application regular operation reestablishment (response time under the maximum limit).

The system parameters are constants. Thus, we keep the exact configuration of VMs, cloud management configurations, and hardware specifications (Section 5.4.4). The cloud and multi-cloud domains are simulated in private environments to avoid distortions on measuring caused by the Internet's speed fluctuations.

The workload parameters (factors) vary to analyse M-CaMid behaviour while it steps in the environment. Section 5.4.4 describes the workload in details. The workload parameters are the following:

- **Number of clients** (NC): number of simultaneous users (client-side instances);
- **Cross management** (CM): it can be turned off, turned on and set to coarse- or fine-grained management ($CM = \{off, cg, fg\}$); and
- **Domain** (D): it indicates the domain where replication takes place (single cloud or multi-cloud) ($D = \{sc, mc\}$).

5.4.3 Treatment

The treatment applied to this experiment is the M-CaMid cross management approach that manages the elasticity at different granularity levels and domains. The absence of treatment is called cross management off in a single cloud domain, while the treatment is called management in the single cloud and multi-cloud domains.

M-CaMid cross management impacts are analysed by comparing multi-cloud application performance without treatment, with M-CaMid cross management acting exclusively at the

infrastructure and application levels. Besides, treatments are compared to validate the use of multiple strategies and domains.

The experiment's control object is a multi-cloud distributed application whose server-side is a remote object that implements the Fibonacci sequence in two situations: without M-CaMid management ($CM = off$) in a single cloud ($D = sc$) and with coarse-grained management ($CM = cg$) in a single cloud ($D = sc$). The experiment submits the object control to a workload to observe its behaviour as the workload varies.

The experimental object is the same as the control object but with M-CaMid management configured to fine-graned management. The experiment also submits the same workload as the control object. Both object behaviours are observed at single cloud and multi-cloud domains.

5.4.4 Experimental Design

A client-server application implemented atop M-CaMid and two cloud technologies, namely OpenNebula (Moreno-Vozmediano et al., 2012) and OpenStack (RackSpace, 2010), were utilized in this experiment. The OpenNebula was configured with an IaaS manager (frontend) and a host for running VMs. The frontend runs on a machine core i5 2.44GHz with 4 GB of memory, and the host is an octa-core Xeon 2.93 GHz machine with 20 GB of memory. Each VM is configured with one CPU core and 1 GB of memory. Three VMs were deployed to execute the experiments: a server-side application and its replicas. OpenStack was configured in a similar way to OpenNenbula.

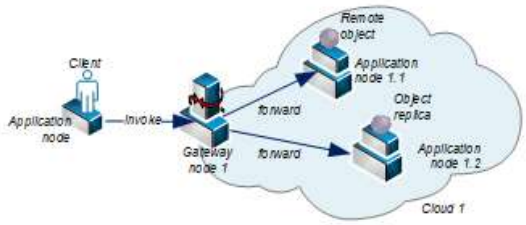
Figure 27 shows the basic configurations used in all scenarios. In the first scenario (S_1), which works as a base case of response time behaviour, cross management is turned off. In Scenario S_2, cross management is set to trigger the coarse-grained strategy, deploying a new VM when necessary. Scenario S_3 describes how existing underused VMs can support new application's replicas without the need for new VM deployments. These three scenarios were deployed in a single cloud environment, as shown in Figure 27a.

In the multi-cloud scenarios (S_4 and S_5), shown in Figure 27b, the current cloud domain do not support more resources to new application's replicas and become necessary to request further resources in another cloud domain. In these scenarios, it is up to the target cloud domain to decide the cross management level that *Adaptor* must trigger. In scenario S_4, the target cloud deploys a new VM (coarse-grained level). In contrast, in scenario S_5, a new application's object replica is created in an existing VM in the target cloud domain.

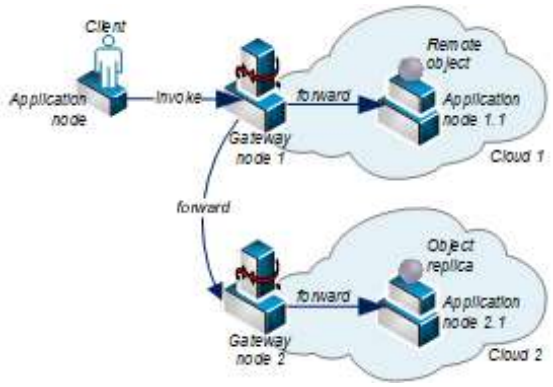
The experiments use a workload pattern having a periodic unexpected workload increase. In this way, it is expected that the response time suddenly increases and exceeds a pre-defined maximum response time. The execution starts with 100 clients that continuously invoke a method to the remote object that runs on the server-side. After 60 seconds, the number of clients increases to 300 and lasts for 200 seconds. During this time, the server-side becomes overloaded, leading to an abrupt increase in the response time. After that, the number of clients decreases to 100, and the response time returns to the same level as before. The same pattern repeats until the

Figure 27 – Basic configurations used in all scenarios.

(a) Single cloud environment.



(b) Multi-cloud environment.



Source: author (2021).

Table 5 – Evaluation scenarios.

Scenario	Cross management strategy	Domain
S_1	OFF	*sc* (single cloud)
S_2	*cg* (coarse-grained)	*sc* (single cloud)
S_3	*fg* (fine-grained)	*sc* (single cloud)
S_4	*cg* (coarse-grained)	*mc* (multi-cloud)
S_5	*fg* (fine-grained)	*mc* (multi-cloud)

Source: author (2021).

end of the experiment.

Each client continuously makes requests, and the time interval between requests has a Gaussian distribution whose mean value and standard deviation are 50*ms* and 12.5*ms*, respectively. Additionally, two thresholds were defined to trigger the creation/release of VMs. If the *response time* becomes higher than 1300*ms* (*highest response time*), a new VM is necessary. Meanwhile, if the *response time* becomes lower than 500*ms* (*lowest response time*), there is an excess of resources being allocated, and a VM needs to be released. All mentioned thresholds were empirically defined through several initial experiments, where the system ran in different scenarios and under diverse operational conditions.

5.4.5 Analysis

This experiment compares the data of experimental and control objects to check the null hypothesis rejection. First, this section discusses the observed scenarios behaviours. It then analyses the cross management impact over the M-CaMid reaction time by denying null hypotheses (Section 5.4.1).Table 6 summarises statistical results of the experiments for each scenario presented in Table 5.

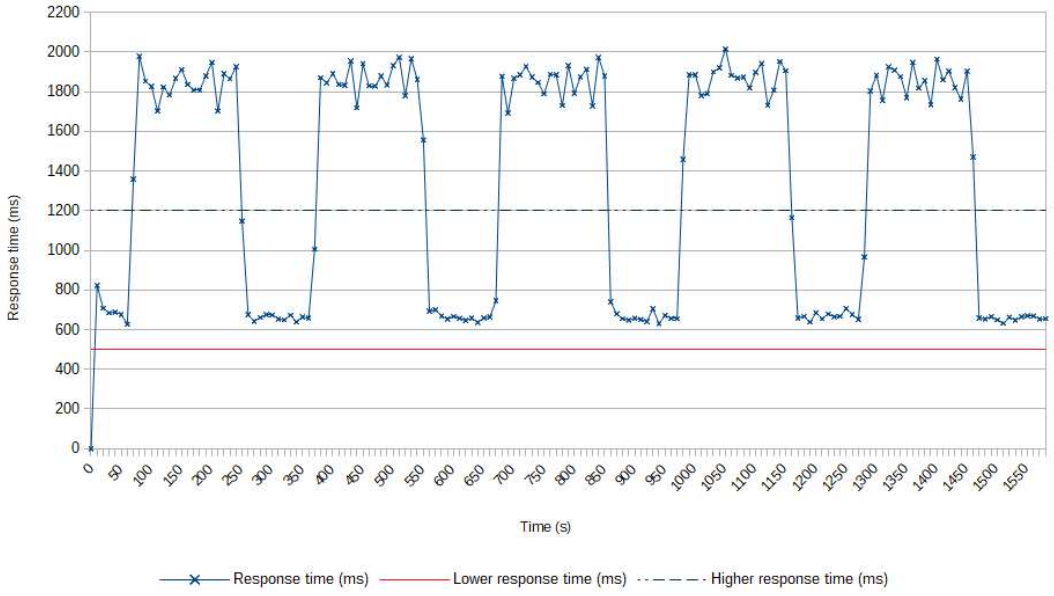
Table 6 – Average reaction time and standard deviation in different experiment's scenarios.

Scenario	Reaction time (s)	Standard deviation (s)
S_1	182	4.472
S_2	110	7.071
S_3	22	4.472
S_4	122	8.365
S_5	36	5.477

Source: author (2021).

Figure 28 shows the behaviour of metric *response time* in scenario S_1. As mentioned before, this scenario works as the base case in which the cross management is disabled, and the *response time* is only impacted by the changes defined by the workload pattern described before.

Figure 28 – *Response time* without elasticity management.



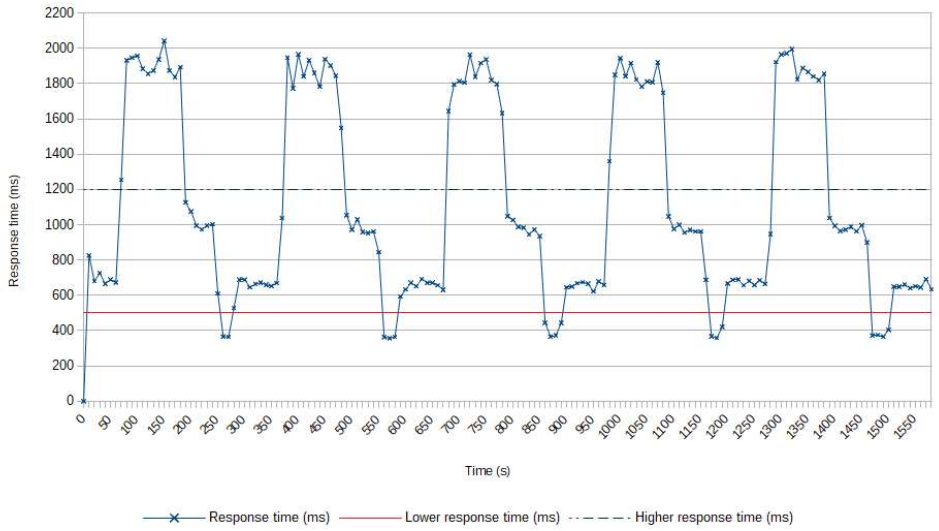
Source: author (2021).

Figure 29 shows the behaviour of *response time* in scenarios in which the M-CaMid cross management is enabled in a single cloud. Figure 29a shows the coarse-grained management level in action as defined in scenario S_2. It is possible to observe that *higher response time* and *lower response time* violations trigger the coarse-grained strategy deploying a new VM to host an object's replica when the *higher response time* is reached. Meanwhile, M-CaMid releases the VM when the *response time* is below *lower response time*. The creation and releasing of VMs occur in a single cloud.

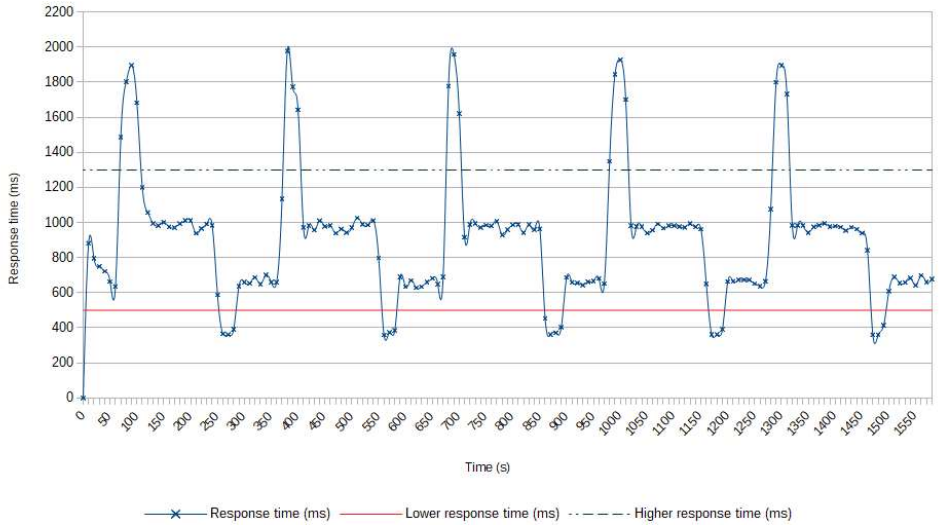
The null hypothesis $H0_1$ proposes no significant difference between the mean response time in S_1 and the mean response time in S_2. In scenario S_1, the meantime between the abnormal event detection and the response time reestablishment was $182s$ with a standard deviation of $4.472s$, while scenario S_2, it was 110 with a standard deviation of 7.071.

Figure 29 – *Response time* with elasticity management (single cloud).

(a) Coarse-grained elasticity.



(b) Fine-grained elasticity.



Source: author (2021).

The null hypothesis $H0_1$ is rejected in the statistical t-test since the result is significant at $p < 0.05$: t-value is 19.243, and p-value is < 0.00001. Thus, the experiment can assert a significant performance gain when M-CaMid manages a multi-cloud application. Furthermore, the effect size for the t-test is Cohen's $d = 12.17$, expressing a hugely significant difference.

In scenario S_3, Figure 29b, M-CaMid steps in the cloud replicating a remote object in an underused VM belonging to the same cloud. In this scenario, it is possible to note the mean response time remains above *higher response time* for a shorter period than in scenario S_2 ($22s$ with a standard deviation of $4.4721s$, and $110s$ with a standard deviation of $7.071s$, respectively). This difference happens because starting a new VM with a coarse-grained management strategy takes longer than finding an existing underused running VM and replicating the remote object.

However, the existence of an underused VM is necessary to execute the fine-grained strategy.

The null hypothesis $H0_2$ proposes no significant difference between the mean response time in S_2 and the mean response time in S_3. In scenario S_2, the meantime between the abnormal event detection and the response time reestablishment was $182s$ with a standard deviation of $4.472s$, while scenario S_2, it was $110s$ with a standard deviation of $7.071s$.

The null hypothesis $H0_2$ is rejected in the statistical t-test since the result is significant at $\alpha = 0.05$: t-value is 19.243, and p-value is < 0.00001. Thus, the experiment asserts a significant performance gain when M-CaMid manages a multi-cloud application. Furthermore, the effect size for the t-test is Glass's delta $\delta = 16.1$, expressing a very significant difference.

The null hypothesis $H0_3$ says no significant difference between mean response time in scenarios S_2 e S_5. The experiment intends to reject this hypothesis to show that the fine-grained management level takes less time even in the multi-cloud domain when compared to the coarse-grained level in a single cloud domain. In Scenarios S_2 and S_5, M-CaMid takes $110s$ ($standarddeviation = 7.071$) and $36s$ ($standarddeviation = 5.477$) to execute the coarse-grained and fine-grained levels, respectively.

The statistical t-test of $H0_3$ results in the hypothesis rejection with $t - value = 18.5$ and $p - value < 0.00001$. The result is significant at $p < 0.05$. It concludes that even if the object replication occurs in another cloud, the taken time to perform the replication is less than deploying a new VM in the same single cloud. The t-test effect size is Cohen's $d = 11.7$, ratifying the null hypothesis rejection.

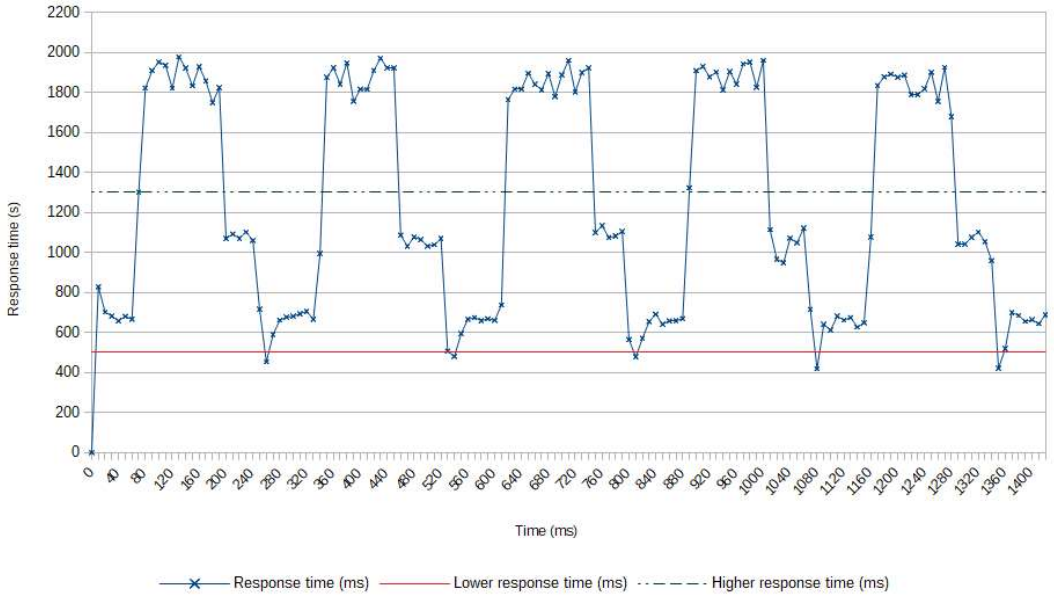
The same comparison between Scenarios S_4 and S_5 is analysed. The null hypothesis $H0_4$ argues no significant difference when M-CaMid cross management performs both management levels in a multi-cloud domain. The t-test result rejects the null hypothesis, presenting a significant difference between the samples at $p < 0.05$ with $t - value = -6.532$, $p - value = 0.000091$. Once more, the result validates that deploying a new VM costs more than replicating a remote object, even in the multi-cloud domain. The Cohen's d effect size reinforces the rejection with $d = 2.8$.

By comparing the behaviour of coarse-grained level in the single cloud (Figure 29a) and multi-cloud (Figure 30a) domains (scenarios S_2 and S_4), it is possible to note that it performs better in a single cloud with *reaction time* of $110s$ (*standard deviation* = 7.072) and $122s$ (*standard deviation* = 8.365) in scenarios S_2 and S_4, respectively, i.e., the coarse-grained management in a single cloud is 10.909% faster than in a multi-cloud environment. This difference has its origin in three facts: different cloud technologies have distinct deployment methods, the VM size differs from cloud to cloud technologies, and the time taken to request, analyse and select another cloud domain. The chosen cloud domain also spends time searching and assigning resources to perform the best-fit approach to meet the new demand.

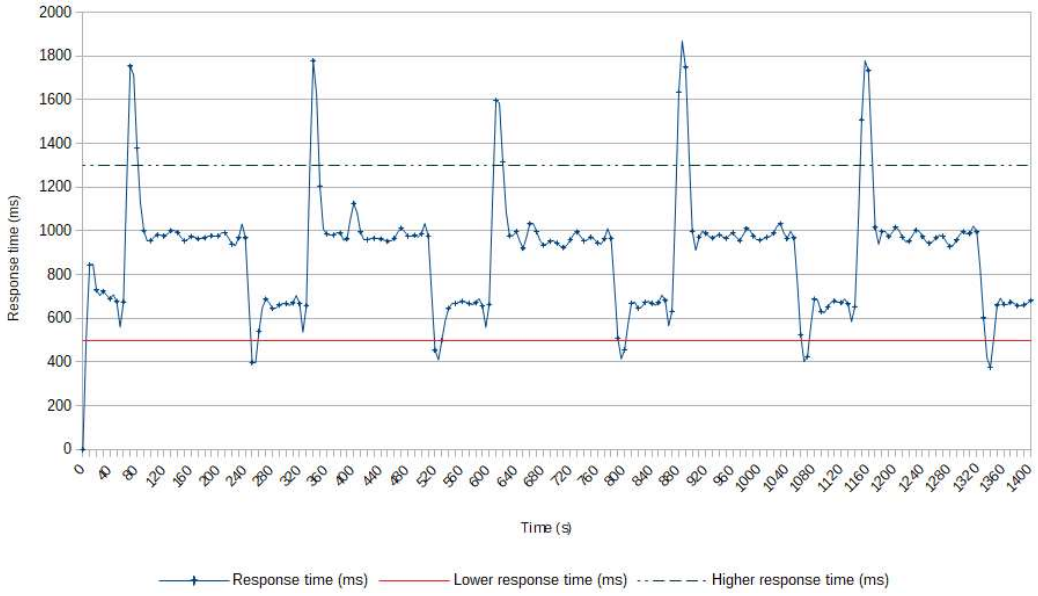
The statistical t-test of $H0_5$ says that there is no significant difference between scenarios S_2 and S_4 results in the hypothesis rejection with $t - value = -2.449$ and $p - value = 0.02$. The result is significant at $p < 0.05$ with a significance level $\alpha = 0.05$. $H0_5$ is rejected with an

Figure 30 – *Response time* with elasticity management (multi-cloud).

(a) Coarse-grained elasticity.



(b) Fine-grained elasticity.



Source: author (2021).

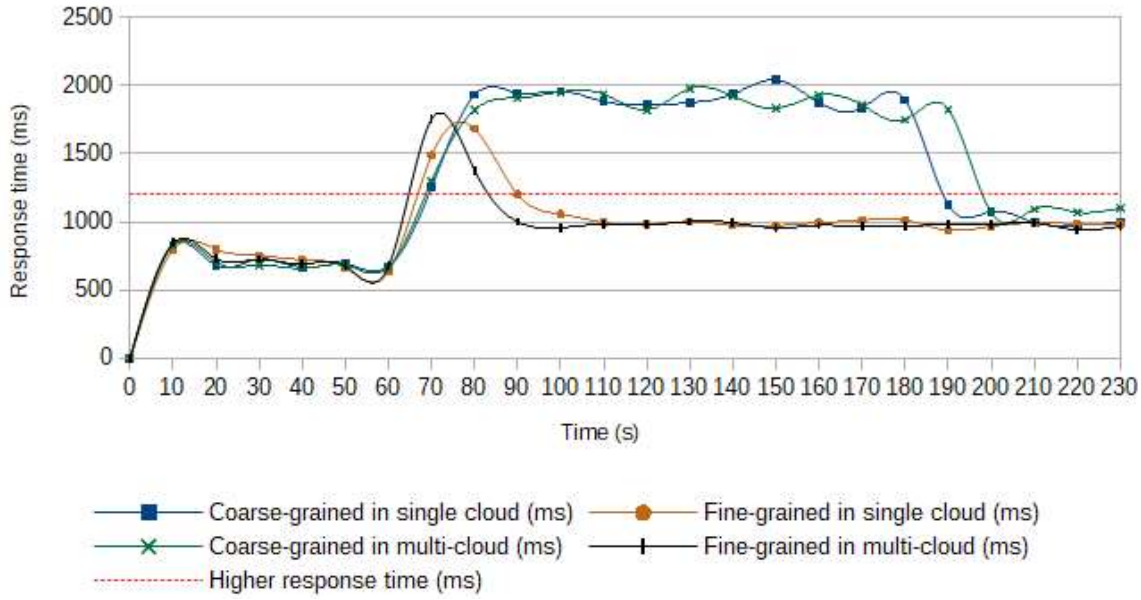
effect size is Cohen's $d = 1.549$, ratifying the null hypothesis rejection. The difference between S_2 and S_4 is because there are delays related to the multi-cloud communication and the cross management actions performed in the chosen cloud domain. However, the effect size is much smaller compared to the other t-tests.

A similar comparison between the performance of the fine-grained level in single (Figure 29b) and multi-cloud (Figure 30b) environments (scenarios S_3 and S_5) shows that it performs 80% better in the single one. Also, this difference is because of the network delay and cross manage-

ment operations delay in the target cloud. These values are absolutes regardless of the domains. Thus, the experiment rejected the hypothesis $H0_6$. t-value is -4.42719 and p-value is 0.001103. The result is significant at $p < 0.05$ with a Cohen's d impact size of $d = 2.8$.

Figure 31 compares the *response time* behaviours in the different scenarios, zooming in a eight minutes interval. In this figure, it is possible to observe how fast M-CaMid cross management reacts when the application is overloaded underloaded, i.e., the response time becomes higher than the *higher response time* and less than the *lower response time*.

Figure 31 – Response time behaviour with elasticity management at different scenarios.



Source: author (2021).

As expected, the fine-grained approach is faster than the coarse-grained one, whatever the environment. The performance of each method is better in the single cloud than in the multi-cloud environment because of network and management delays performed in the case of two single cloud negotiation and management (multi-cloud domain). Table 7 summarises the null hypothesis statistical t-test values.

Table 7 – Experiment t-tests results.

Hypothesis	**Null hypothesis**	***t-value***	***p-value***	**Cohen's *d***
$H0_1$	rejected	19.243	< 0.00001	12.17
$H0_2$	rejected	19.243	< 0.00001	16.1
$H0_3$	rejected	18.5	< 0.00001	11.7
$H0_4$	rejected	-6.532	< 0.00001	12.162
$H0_5$	rejected	-2.449	0.02	1.549
$H0_6$	rejected	-4.42719	0.001103	2.8

Source: author (2021).

It is worth observing that although rejection of the null hypotheses $H0_5$ and $H0_6$, Cohen's

impact sizes are too small compared to the other ones. It means that both null hypotheses t-test are much closer to no rejection than the other hypotheses analysis results.

5.5 Resource Management

One of the M-CaMid goals is the rational usage of infrastructure resources through the cross management method. Resource management experiment assesses if M-CaMid reaches its goals by observing how management leverages infrastructure to maximise resource usage.

It is expected M-CaMid leverages underused VMs (fine-grained management - fg) instead of creates new VMs (coarse-grained control - cg). In this scenario, there are underused VMs in cloud domain which M-CaMid must analyse the possibility of object replication instead of new VMs deployments. The resource usage is evaluated through metrics related to VM processor usage.

The experiment evaluates the amount of VMs and the CPU percentage used by the server-side application in two scenarios: when M-CaMid cross management is configured with coarse-grained (S_1) and fine-grained (S_2) levels. Results of both scenarios are compared to decide which one is more efficient. Furthermore, the experiment calculates the efficiency of M-CaMid using infrastructure resources EFF (Equation (1)).

5.5.1 Hypotheses

Based on the previous description, The experiment can define its hypotheses. Thus, null hypotheses advocate no difference between the M-CaMid cross management strategies regarding resource usage:

$H0 : CPU\_usage_{cg} \cong CPU\_usage_{fg}$

Alternative hypotheses argues that fine-grained level uses better available resources than the coarse-grained level:

$H1 : CPU\_usage_{cg} < CPU\_usage_{fg}$

5.5.2 Metrics, Parameters and Factors

The metric adopted to assess resource usage efficiency (EFF) is the proportion of resources used and resources available – Equation (1). The resource usage is measured in CPU percentage with value in a 0..1 range. Furthermore, CPU usage is related to the mean application's response time. Even if the CPU usage is at 1 (100%), the average response time can be within the pre-defined limits. It means that, probably, the application is using the CPU as much as possible. On the other hand, if the average response time increases out of bounds while the CPU usage remains at 1, it means a performance depreciation. The mean response time (RT) is measured in seconds.

$$EFF = \frac{\sum CPU\_usage}{\sum CPU\_available} \tag{1}$$

The system parameters are constants. Thus, we keep the exact configuration of VMs, cloud management configurations, and hardware specifications (Section 5.4.4). The cloud domain is simulated in private environments to avoid distortions on measuring caused by the Internet's speed fluctuations. Besides, available underused resources are set with a small CPU workload (≈ 0.25).

The workload parameters (factors) vary to analyse the resource usage by M-CaMid while it steps in the environment. Section 5.5.4 describe in details the workload. The workload parameters are the following:

Number of clients - NC : number of simultaneous users (client-side instances); and

Cross management - CM : it is set to coarse-grained and fine-grained management levels ($CM = cg, fg$).

5.5.3 Treatment

The treatment applied to this experiment is the M-CaMid cross management approach that manages the elasticity at different granularity levels. The reference treatment is the coarse-grained level in a single cloud domain, while the treatment is the fine-grained level in a single cloud domain.

M-CaMid cross management fine-grained level impacts on the resource usage are assessed by comparing to the coarse-grained level. Coarse-grained management is the usual elasticity level adopted in cloud providers.

The experiment's control object is a multi-cloud distributed application whose server-side is a remote object that implements the Fibonacci sequence in two situations. M-CaMid is set to coarse-grained level ($CM = cg$) in a single cloud. The experiment submits the object control to a workload to observe its behaviour as the workload varies and triggers cross management.

The experimental object is the same as the control object but with M-CaMid management configured to fine-grained management. The experiment also submits the same workload as the control object. Both objects behaviours are observed at a single cloud.

5.5.4 Experimental Design

The experiment system follows the same configuration as defined in Section 5.5.4. Each VM is configured with one CPU core and $1GB$ of memory. Three VMs were deployed to execute the experiments: a server-side application and its replicas. One of the VMs is the available underused resource (AR), with a Gaussian distribution whose average workload of 0.25 and a standard deviation of 0.05.

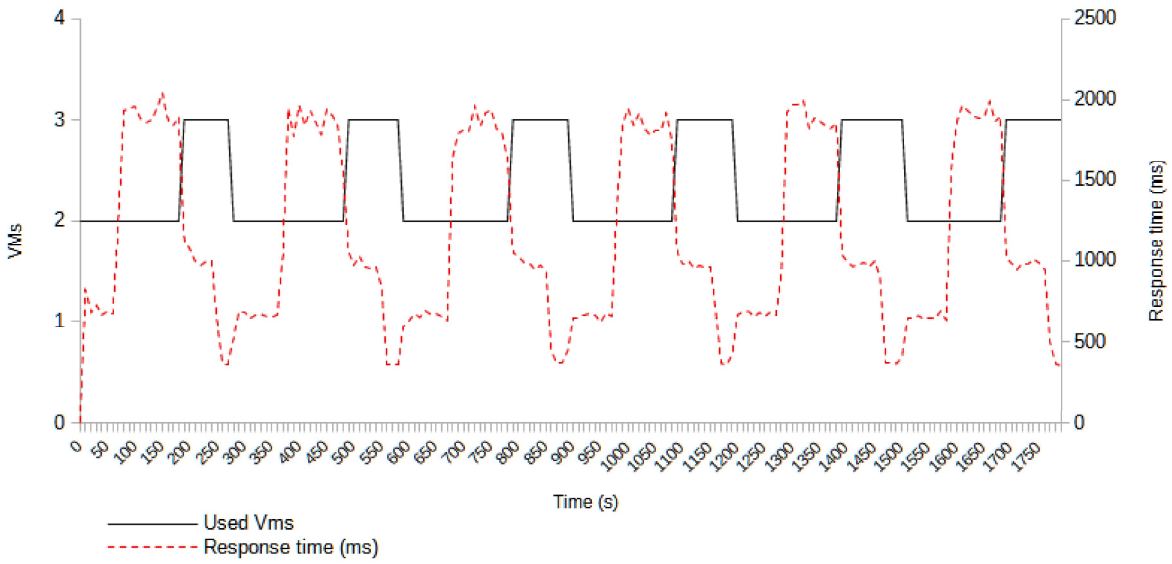
The number of clients (NC) increases and decreases to generate different workloads to trigger cross management control to step in the environment and observe the relation between used resources and available resources.

The experiment starts with 100 clients invoking remote objects. Each client continuously invokes a remote object in time intervals of a Gaussian distribution whose mean value and standard deviation are 50*ms* and 12.5*ms*, respectively. Additionally, two thresholds were defined to trigger cross management. If the *response time* becomes higher than 1300*ms* (*highest response time*), new resource is necessary.

5.5.5 Analysis

The analysis discusses the resource usage during the experiment by comparing it with the application response time measurement in scenarios S_2 and S_3. Figure 32 shows resource usage by M-CaMid coarse-grained management. The graph overlaps the measurements, showing when M-CaMid add a new node, the application response time decreases. Similarly, when the response time is under the lower limit, the node is put away, increasing the response time again but still within limits.

Figure 32 – Node usage in coarse-grained strategy.

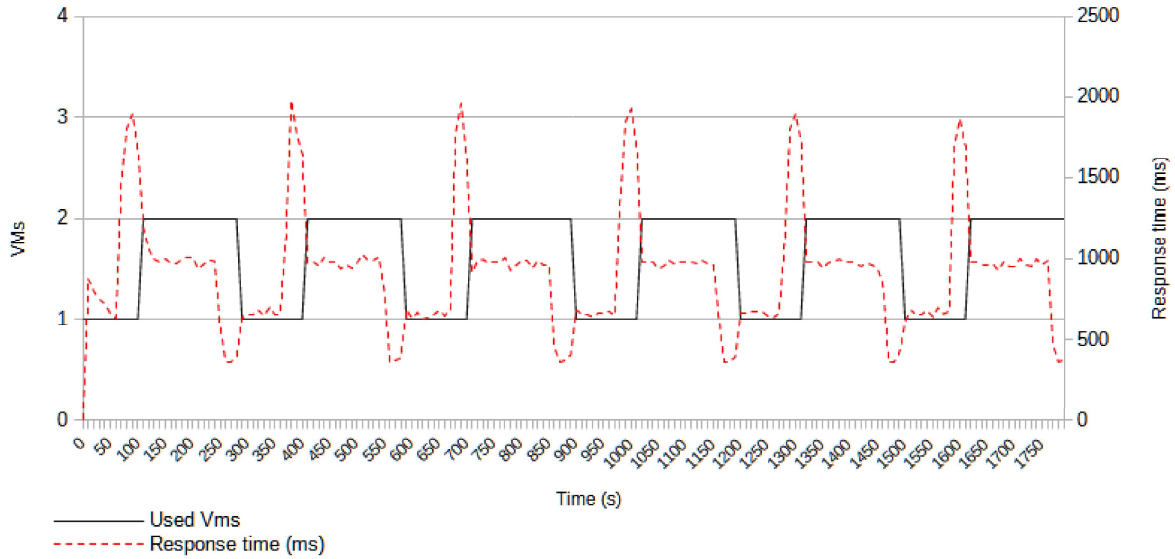


Source: author (2021).

The same behaviour occurs when M-CaMid executes fine-grained management. However, the reaction time is shorter, and the amount of the used node is smaller. Figure 33 shows used nodes by fine-grained strategy along the experiment time. A fine-grained approach takes advantage of underused resources.

Regarding CPU usage, Figure 34 shows available CPU percentage (dashed line) and used CPU by M-CaMid coarse-grained strategy. Note that there are no utilising resources as occurs in traditional infrastructure elasticity. Instead, it deploys new resources regardless of underused resource availability. On the other hand, M-CaMid fine-grained strategy leverages underused resources to avoid resource-wasting, as shown in Figure 35. M-CaMid keeps a rational number

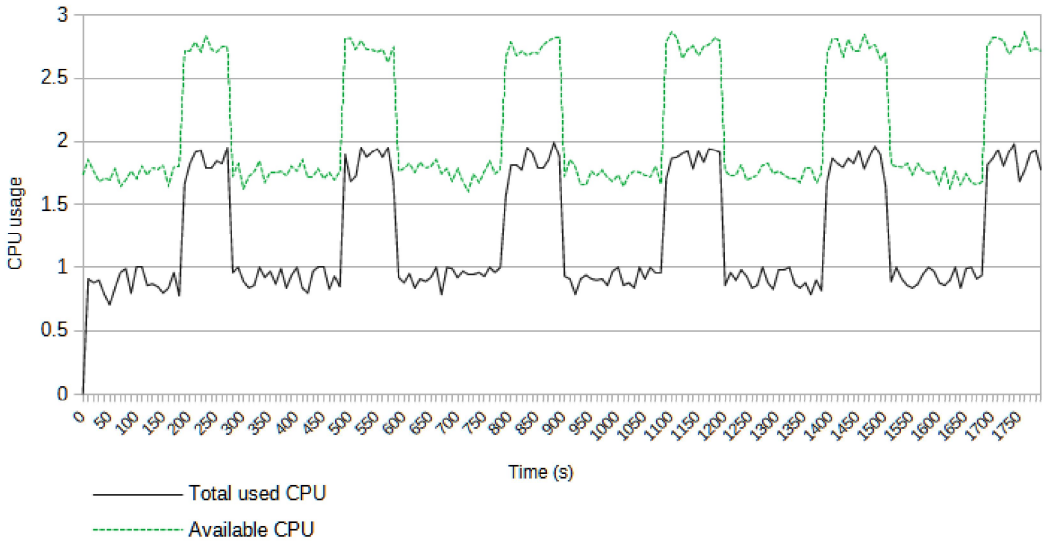
Figure 33 – Node usage in fine-grained strategy.



Source: author (2021).

of nodes whenever possible.

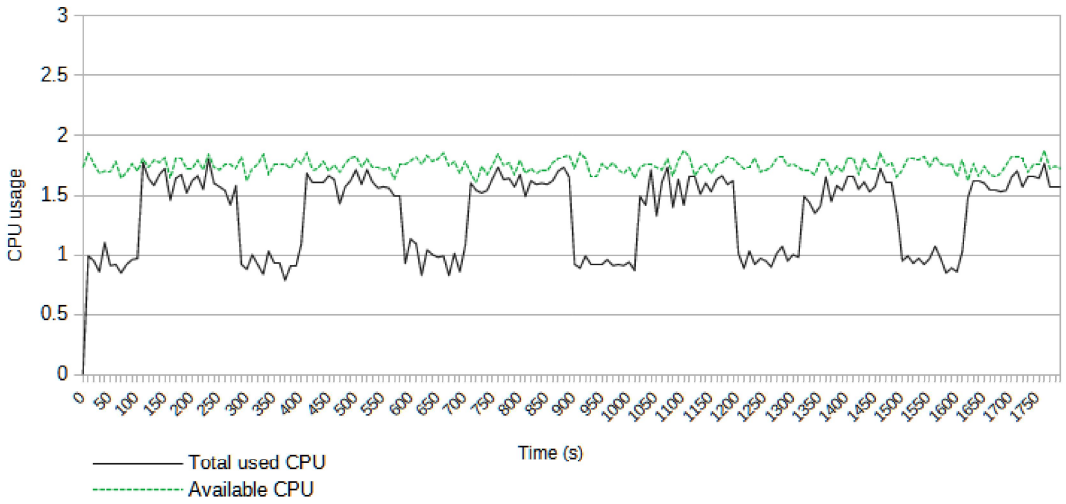
Figure 34 – CPU usage in coarse-grained strategy.



Source: author (2021).

The CPU usage average in both strategies is presented in Figure 36. It compares the total used and total not used CPU by the server-side application, considering coarse-grained and fine-grained cross management levels. Coarse-grained wastes about 0.532 of 3 CPUs, and fine-grained does not use only 0.08 from 2 VMs. The coarse-grained efficiency is $EFF = 0.194$ and fine-grained has $EFF = 0.954$. Table 8 summarises the results of rational usage of CPU resources.

Figure 35 – CPU usage in fine-grained strategy.



Source: author (2021).

Table 8 – CPU usage efficiency (EFF) in coarse and fine-grained management strategies (scenarios S_1 and S_2, respectively).

Scenario	Adaptation strategy	Underused CPU ratio	Used CPU ratio	Available CPU ratio	*EFF*
S_2	cg (coarse-grained)	2.204	0.532	2.736	0.194
S_3	fg (fine-grained)	0.08	1.656	1.736	0.954

Source: author (2021).

With these results, the null hypothesis $H0$ can be rejected because the result is significant at $p < 0.05$ with $t - value = -6.833$ and $p - value = 0.000067$. The effect size Cohen's $d = 4.49$. Thus, the experiment shows the efficiency of M-CaMid using cloud resources is satisfactory.

5.6 Performance overhead

This experiment assesses the overhead introduced by M-CaMid in the multi-cloud application performance and its infrastructure. It intends to observe the impact on invocations' response time when the experiment submits a multi-cloud application to a variable workload. This investigation also defines levels of workload supported by M-CaMid.

As outcomes, it is expected M-CaMid manages remote objects and their infrastructures during the multi-cloud application execution without significant overhead introduction. The overhead can be expressed by the metric response time measured when the client-side application invokes a successively remote object (*RO*). This metric evaluates the overhead over the multi-cloud application. The infrastructure overhead is expressed by VM CPU usage (*VM*). The experiment occurs in two scenarios: M-CaMid with cross management turned OFF (*OFF*) and turned ON (ON).

Figure 36 – CPU percentage used by M-CaMid cross management.

Source: author (2021).

5.6.1 Hypotheses

In the experiments null hypotheses ($H0_{RO..VM}$), there are a significant difference among response times ($H0_{RO}$) and significant differences among CPU usage ($H0_{VM}$) when M-CaMid management is OFF or ON.

$H0_{RO} : RO_{OFF} < RO_{ON}$

$H0_{VM} : VM_{OFF} < VM_{OFF}$

On the other hand, the alternatives hypotheses ($H1_{1..3}$) are:

$H1_{RO} : RO_{OFF} \cong RO_{ON}$

$H1_{VM} : VM_{OFF} \cong VM_{OFF}$

5.6.2 Metrics, Parameters and Factors

As aforementioned, response time and CPU usage are metrics to access how M-CaMid management affects multi-cloud applications and VM performances, respectively, by introducing workload overhead.

The system parameters are fixed, keeping the exact configuration of VM, cloud management configurations, and hardware specifications (Section 5.4.4). The experiment carries only on the cloud domain since it is expected that VMs have similar behaviour regardless of the cloud

technology. Furthermore, M-CaMid switches off parameters for triggering actions because the experiment aims to stress the system to observe the elements' behaviours.

Some factors affect the M-CaMid performance. The workload parameters (factors) vary to analyse M-CaMid behaviour. They are:

1. **Number of clients**: number of simultaneous users (client-side instances);
2. **Number of invocations**: number of successive remote invocations performed by each user;
3. **Cross management**: it can be turned on or turned off.

Table 9 shows factors and their respective values assumed in the experiment.

Table 9 – Factors and values of M-CaMid overhead.

Factor	Values
Simultaneous clients	$100, 200, 300, 400, 500, 600, 700, 800, 900, 1000$
Invocation interval	Gaussian distribution: mean value $100ms$ standard deviation $12.5ms$
Cross management	turned off and turned on

Source: author (2021).

5.6.3 Treatment

The treatment applied to this experiment is the M-CaMid approach that monitors and controls the multi-cloud application environment. The absence of treatment is called cross management off (OFF), while the treatment is called management on (ON).

M-CaMid cross management impacts are analysed by comparing response time measures of a multi-cloud application under different workload levels with (ON) or without (OFF) treatment.

The experiment's control object is a multi-cloud application whose server-side is a remote object that implements the Fibonacci sequence without M-CaMid cross management (OFF). The experiment submits the object control to a workload to observe the system behaviour as the workload varies.

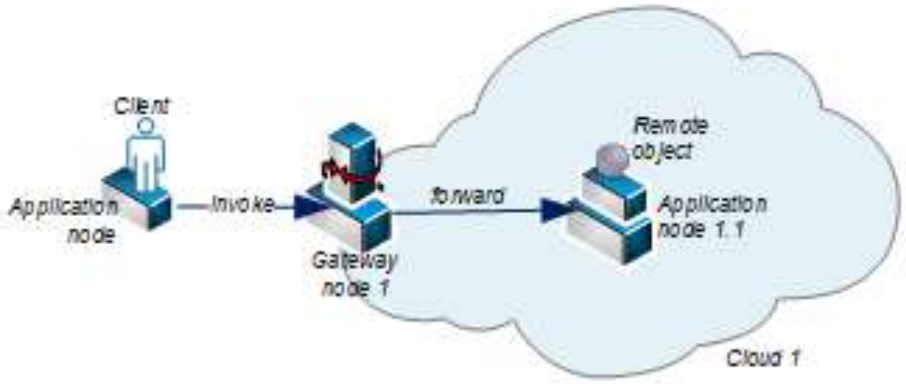
The experimental object is the same as the control object but with M-CaMid management turned on (ON). The experiment also submits the same workload as the control object. Both objects behaviours are observed at a single cloud domain since the same behaviour is expected regardless of cloud technology.

5.6.4 Experimental Design

Figure 37 shows the basic configurations used in the experiment. The experiment system comprises the client-server application implemented atop M-CaMid and OpenNebula (Moreno-Vozmediano et al., 2012) as cloud infrastructure. The OpenNebula was configured with an IaaS

manager (frontend) and a host for running VMs. Frontend runs on a machine core i5 2.44GHz with 4 GB of memory, and the cloud host (where the frontend deploys VMs) is an octa-core Xeon 2.93 GHz machine with 20 GB of memory. Each VM is configured with one CPU core and 1 GB of memory.

Figure 37 – M-CaMid scenario for M-CaMid overhead experiment.



Source: author (2021).

Both M-CaMid's monitors (node and cloud) are configured to gather monitoring data in an interval of 10*s*. Common real-world monitoring systems time intervals include 1, 5, 15, and 60 minutes. The experiment set M-CaMid time interval as 10*s* because its goal is to observe M-CaMid's behaviour under stressful workloads. Furthermore, in the experimental object, the thresholds (lowest and highest response times) are set with values out of the range to prevent M-CaMid management from adding or removing infrastructure resources and biasing the experiment result.

The experiment submits M-CaMid to a growing workload. The number of users increases, assuming values as described in Table 9. The M-CaMid behaviour is observed through response time (milliseconds) and throughput measurements (request per second).

5.6.5 Analysis

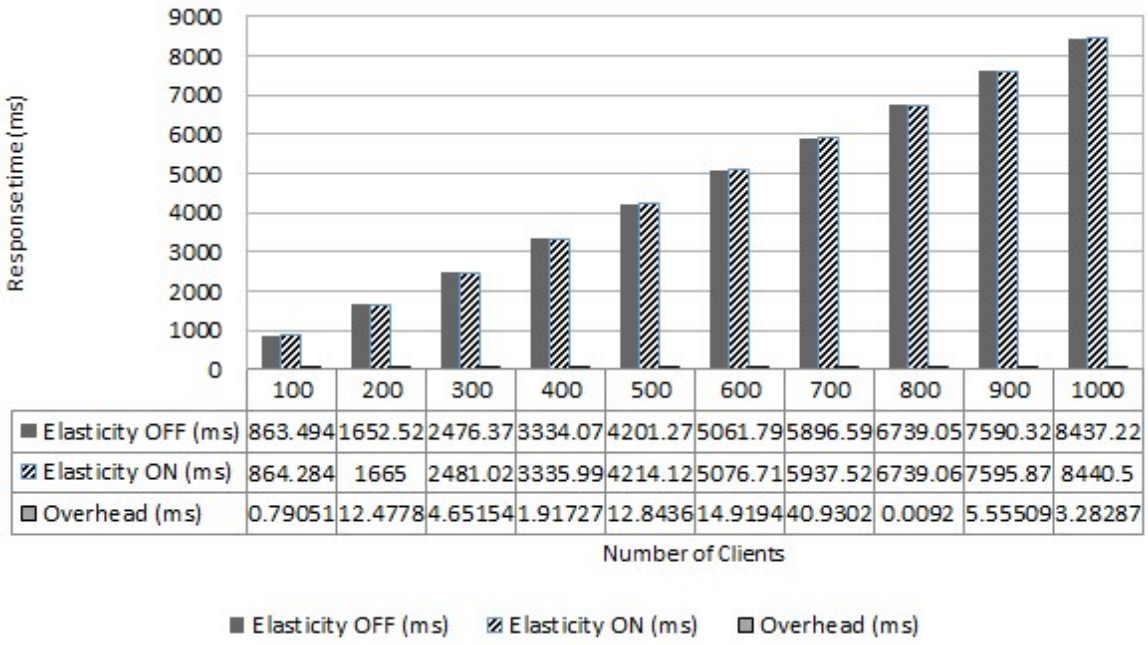
The study analysis compares the experimental object's measurements and control object trials to reject the null hypotheses. The experiment assesses the M-CaMid performance impact over the multi-cloud application response time.

Section 9 described the applied factor values. A t-test compares the mean values of two data sets: with M-CaMid management turned off and with M-CaMid management turned on. T-test analyses if their differences are not significant.

The experiment goal is not to reject the null hypothesis since the experiment intends to show no significant difference between the means. The null hypothesis claims the mean response times are equivalents with M-CaMid management ON than with M-CaMid management off ($H0_{RO} : RO_{OFF} < RO_{ON}$). The experiment calculates the mean response time with the two M-CaMid configurations for each number of clients. The assumed test's confidence is 95%.

Figure 38 shows the mean response time when the M-CaMid management is ON and OFF. As the application is the same in both cases, the difference between mean response times is the overhead of the M-CaMid management. The mean overhead is 3.067% even when the application server is submitted to a heavy workload, i.e., 1000 clients. The lowest and highest overheads are 1.626% and 5.603%, respectively. In this way, it is intuitive to observe that cross management has a minimal impact on the multi-cloud application's performance.

Figure 38 – M-CaMid overhead impact over application response time.



	100	200	300	400	500	600	700	800	900	1000
Elasticity OFF (ms)	863.494	1652.52	2476.37	3334.07	4201.27	5061.79	5896.59	6739.05	7590.32	8437.22
Elasticity ON (ms)	864.284	1665	2481.02	3335.99	4214.12	5076.71	5937.52	6739.06	7595.87	8440.5
Overhead (ms)	0.79051	12.4778	4.65154	1.91727	12.8436	14.9194	40.9302	0.0092	5.55509	3.28287

Source: author (2021).

Regarding the statistical test, Table 10 shows the results. The t-value is -0.00745 and the p-value is 0.497065. The result (mean difference) is not significant at $p < 0.05$. The null hypothesis can not be rejected.

Table 10 – T-student test statistics values.

α	**Difference of means**	**p-value**	**t-value**
0.05	-8.85	0.497065	-0.497065

Source: author (2021).

5.7 Concluding Remarks

This chapter presented the evaluation of M-CaMid. The primary goal was to show the performance of M-CaMid supporting the execution of multi-cloud distributed applications. A systematic approach to performance evaluation was adopted to accomplish the experiments. Initially, some steps were defined to conduct the experiments: goals, hypotheses, scenarios, metrics, parameters and factors, and experiment design. Finally, results were presented and discussed through statistical tests.