Benchmarking Scalability of Message Transport Systems in the JADE Platform: Experimental Evaluation and Performance Analysis

Luis Felipe Ferin Sgursky, Arthur Casals, Anarosa Alves Franco Brandao

[1]Escola Politécnica - Universidade de São Paulo (EPUSP)
Av. Prof. Luciano Gualberto, 158 - trav.3 - 05508-900 - São Paulo - SP - Brazil

Abstract. *Agents are distinct entities known for their independence and autonomy. In a multi-agent system, multiple agents can interact with each other. In this context, communication is a crucial component in enabling relationships between agents, and it is one of the fundamental features provided by a multi-agent system platform. As such, the performance of a multi-agent system can be directly affected by the implementation of its communication mechanism. In this paper, we analyze the scalability of the JADE platform from the perspective of its communication mechanism. We do this by defining benchmarks and evaluating the platform's response in different scale-up and scale-out scenarios.*

1. Introduction

Agents, as relatively independent and autonomous entities, are meant to solve problems of varying complexity [Ferber 1999]. There are different types of agents, ranging from reactive agents to those capable of intelligent reasoning, such as deliberative agents [Balke and Gilbert 2014]. A BDI agent is a particular type of deliberative agent that makes decisions similarly to humans [Thangarajah et al. 2002], based on beliefs, desires, and intentions. Other types of agents include hybrid agents, which combine the strengths of reactive and deliberative agents. With their overall interactive and social capabilities, agents serve as a paradigm for software engineering [Jennings 2000].

Multi-agent systems integrate software agents that collaborate to achieve goals, using different interaction mechanisms [Stone and Veloso 2000, Russell and Norvig 2016]. One of the purposes of a MAS is to autonomously solve complex problems that are challenging for monolithic systems. They reduce system complexity and coupling between components, thus being capable of adapting to unexpected conditions.

Software agents possess communication skills for data and message sharing [Russell and Norvig 2016]. Agents communicate among themselves through the use of agent communication languages (ACLs) [Jennings 2000, Bellifemine et al. 2001]. However, communication between MASs can face complexity and interoperability limitations [Poslad and Charlton 2001].

This paper presents an experimental study that aims to evaluate the scalability of the JADE platform, with a particular focus on its communication mechanism. The goal is to evaluate how the platform performance is affected when increasing communication components, such as the number of agents, the number of messages sent, and the size of the messages. By systematically varying these scenarios and measuring the related performance metrics, it is possible to analyze patterns that indicate characteristics of the platform's scalability.

This paper is structured as follows: Section 2 provides a brief overview of related work on scalability for MAS platforms. Section 3 presents the methodology chosen to perform the experiments in this paper. Section 4 details each experiment in depth. Section 5 presents the results of each experiment and its correlation with the scalability of the platform. In section 7 we discuss the results obtained from the experiments.

2. Related Work

In [Vitaglione et al. 2002], the authors evaluated the communication performance of the JADE platform by measuring the duration of a conversation between agents in two distinct situations: agents deployed into the same platform (intra-platform), and agents deployed into different platforms (inter-platform). Also, there were two distinct intra-platform scenarios: (i) agents deployed within the same container, and (ii) agents deployed into different containers. These tests, however, were focused on analyzing the communication implementation in JADE's communication middleware, attributing the results obtained to the higher level of abstraction required by ACL-compliant communication.

In [Such et al. 2007], the authors used a different methodology from [Vitaglione et al. 2002], using a fixed number of communication couples in each experiment (instead of gradually increasing their numbers). They proposed benchmark metrics to analyze the performance impact of scaling the communication between agents in different aspects, such as increasing the number of hosts, massive reception of messages from one host, and scaling the number of agents within a platform. Those experiments provide a higher level of abstraction because they evaluate communication aspects from MAS in general, instead of aiming for a specific platform. This allows different MAS platforms to be compared using the same methodology and metrics.

In another work, [Rodrigues 2019] provided a practical implementation and analysis between several MAS platforms such as JADE, SPADE, JIAC V, and ASTRA. More recently, [Alencar 2020] expanded this work using instances of the JADE platform deployed into the cloud, using on-demand Amazon Elastic Compute Cloud (Amazon EC2) services.

Our work extends the work done by [Rodrigues 2019] and [Alencar 2020] in terms of scalability. The communication metrics are the same as used for [Vitaglione et al. 2002], but the benchmarks chosen to evaluate the platform are the ones defined by [Such et al. 2007].

3. Context

3.1. Introduction

In distributed systems, scalability is an important aspect from the perspectives of performance, responsiveness, and reliability. Similarly, in a MAS, the system must be able to handle the communication between the agents efficiently, since increasing the number of agents in a system will also increase the interactions among them. As we mentioned before, our objective is to study the scalability of a MAS built using the JADE platform, focusing on its communication mechanism.

There are many different ways in which a system can be scaled. For the purposes of this work, we will focus on two different scaling approaches: *scaling up*, referring

to adding more resources (e.g. memory, processor) to a system already deployed, and *scaling out*, which is when we deploy more instances of a system (so it can handle a greater workload). For our study, we analyze how a MAS built using JADE responds to both approaches.

3.2. JADE

JADE (Java Agent Development Framework) [Bellifemine et al. 2007] is a decentralized FIPA-compliant, Java-based agent platform, used for the implementation of agents and MAS. The platform runs on the JVM (Java Virtual Machine) and hosts agents in containers, each being a Java process. The platform is capable of hosting containers within the same host or even hosts spread across the web. It also supports inter-platform communication, where agents deployed on different platforms can communicate with each other. To enable agent communication, JADE uses MTPs (Message Transfer Protocols) that allow communication between agents. The protocol and implementation used depend on the location of the agents [Bellifemine et al. 1999].

When dealing with intra-platform communication, if all agents are within the same container the communication is handled by IMTP (Internal Message Transfer Protocol). If the agents are located in different containers, the communication will be handled through RMI (Remote Method Invocation). In an inter-platform scenario, communication can be established via many different protocols. In our experiments, we used IMTP for intra-platform communication (since all agents were deployed in the same container) and HTTP MTP (MTP based on HTTP) for inter-platform communication.

3.3. Architecture and Environment

The MAS evaluated in this work consists of multiple JADE platforms running within a cluster. We use two different configurations for our experiments: when measuring benchmarks 1 to 3 (explained below), each JADE platform has only the main container, and all of its agents are registered under it. When measuring benchmark 4, each JADE platform has multiple containers. The details of each configuration are shown in the next section (Figures 2 and 3).

All infrastructure used in our experiments was cloud-based. We used AWS (Amazon Web Services) as our main cloud infrastructure provider. Each JADE platform was instantiated in a dedicated Docker [1] container, thus providing a high level of isolation between the deployed platforms. This setup can be seen in Figure 1

Using clusters to deploy the MAS was the approach we chose to overcome the limitations faced by [Alencar 2020]: the deployment environment was limited to 32 Amazon Elastic Compute Cloud (EC2) instances, and it was also necessary to manually deploy and setup each instance. By using a cluster with Amazon Elastic Container Service (ECS) as the orchestration mechanism, the limit of instances running goes from dozens to thousands, and the allocation of the JADE platforms within the docker containers is optimized across the cluster. This allows us not only to greatly increase the number of JADE hosts but also to better understand the communication constraints in the JADE platform within a highly scalable scenario.

[1] https://www.docker.com/

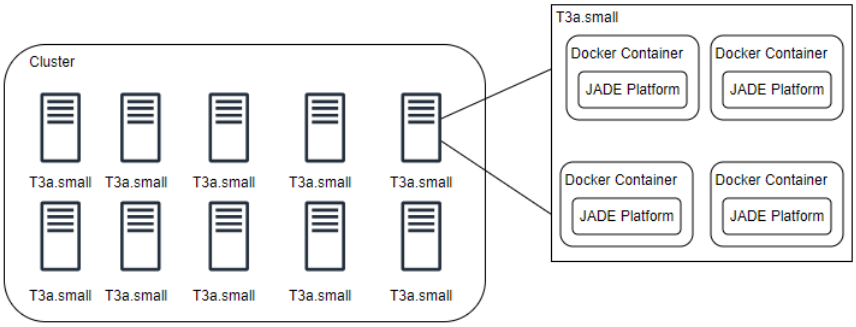

Figure 1. Cluster

The cluster is composed of multiple EC2 T3a.small instances with the default hardware configuration and networking settings [2]. The instances were created on demand by the orchestrator, and the resources used were kept during the experiments, thus maintaining the server performance stable across the experiments. By using the on-demand proposition on AWS, once the cluster is created, we guarantee that the created resources will be available on the cluster, removing the risk of impact on the test performance due to external factors (such as dynamic reallocation of resources by the cloud provider).

In this experiment, each cluster instance hosted multiple docker containers. Each of these containers was hard-limited to allocate a maximum 512MB of memory and 0,5 VCPU on the scale-out experiments (equivalent to 25% of the EC2 instance's hardware resources). For scale-up experiments, the allocation of Memory and VCPU per docker container varied.

Each agent can perform two different roles: sender and receiver. Sender agents are responsible for sending a message, waiting for its reply, and storing the round trip time. Receiver agents will wait for messages and reply to them. Communication can be established between agents in the same JADE platform (intra-platform) and agents in different platforms (inter-platform).

4. Experiments

The experiments were evaluated by measuring the communication performance between agents. The metric used to analyze the performance is the average round trip time (RTT) of the messages exchanged between two agents (Sender and Receiver). The Sender will start the communication and send a message to the Receiver, measuring the time it will take to receive a reply. The average RTT (avgRTT) is a well-known metric, commonly used on network communication-related benchmarking [Tanenbaum 2003]

4.1. Scale-Out

In the context of this paper, the scale-out approach refers to increasing the number of software instances running, in our case, JADE platforms. To evaluate the scalability of the platform using the scale-out approach, we adopted a set of four benchmarking scenarios (as defined by [Such et al. 2007]), which allowed the evaluation of different layers

[2]https://aws.amazon.com/ec2/instance-types/

of the platform communication mechanism. We will refer to these scenarios simply as ”benchmarks.” The first benchmark evaluates how the platform is affected as the multi-agent system grows and has more agents interacting with each other, providing a general response to the platform’s scalability.

The second benchmark involves concentrating all the messages being sent to a single receiver agent. In this context, we can evaluate how the platform handles the massive receive of messages from a single agent’s perspective. The third benchmark expands the previous benchmark by adding multiple receiver agents in the same platform. The total number of received messages is still the same, but as the messages are received by different agents, using both benchmarks allows us to evaluate the limitations of the platform when receiving multiple messages and how its performance is affected when multiple agents are responsible for handling the messages (in contrast to a single agent).

The fourth benchmark evaluates the platform response in dealing with inter-platform and intra-platform communication, and how the messages exchanged between agents in different platforms affect the average RTT.

In our experiments, all benchmarks are evaluated against the same parameter variations, the number of senders(NS), the number of messages(NM), and the message size(MS).

4.1.1. Scale-Out Benchmark 1: Number of Hosts

In this benchmark, each host platform contains one Sender and one Receiver. The Senders send messages to all Receivers allocated in different JADE platforms. In this scenario, the total number of exchange messages is NS*(NS-1)*NM. An example of this benchmark is shown in Figure 2.

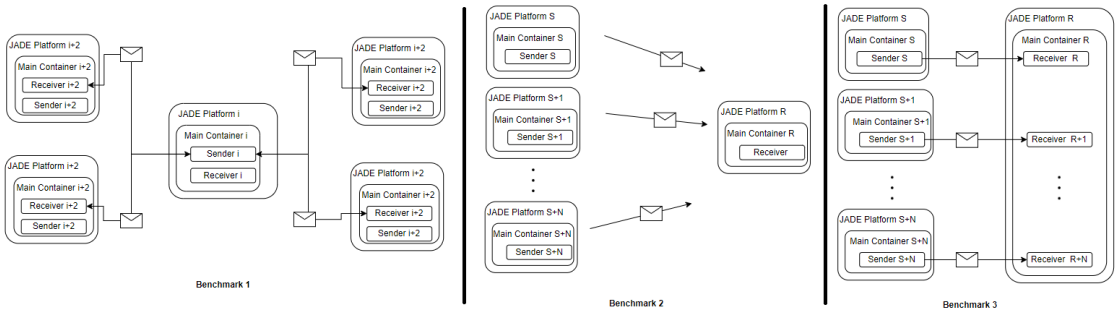

Figure 2. Benchmarks 1, 2 and 3

4.1.2. Scale-Out Benchmark 2: Massive Reception on Single Receiver

This benchmark analyzes how the platform behaves if a single Receiver is in charge of receiving and replying to all the messages originating from the Senders. For this benchmark, the total number of received messages by each receiver agent is NS*NM, as shown in Figure 2.

4.1.3. Scale-Out Benchmark 3: Massive Reception on Multiple Receivers

This benchmark complements the previous one by adding multiple Receivers to handle the messages. All messages are still sent to the same host, but now the workload to receive the messages is divided by several agents across the platform. The host still receives NS*NM messages, but every agent receives only NM messages since every Sender only sends messages to one Receiver agent during the experiment. A diagram of this flow is presented in Figure 2.

4.1.4. Scale-Out Benchmark 4: Number of Agents per Host

The last scale-out benchmark analyzes the platform performance when dealing with multiple agents exchanging messages on it. This benchmark analyzes the effect on the average RTT in two different communication scenarios: intra-platform and inter-platform. For intra-platform communication, all agents are placed on the same platform, and message exchanges follow the 1:1 relationship between sender and recipient, as shown in Figure 3.

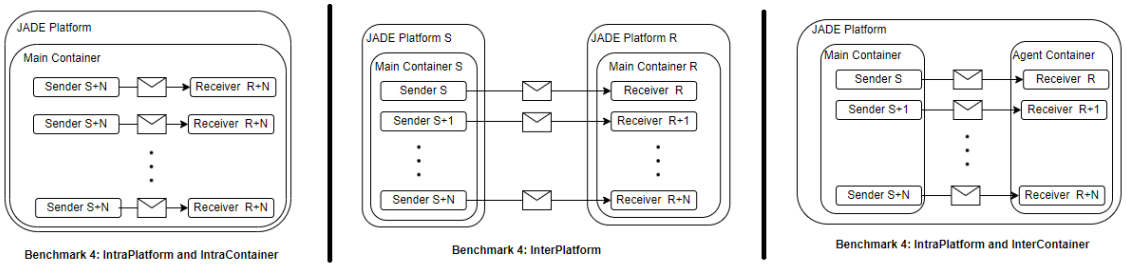

Figure 3. Benchmark 4

For inter-platform communication, the Senders are placed on one platform and the Receivers on another platform. The communication pattern is the same as the one used for intra-platform communication: the Senders only send messages to a unique Recipient, as shown in Figure 3. Also, as explored in [Vitaglione et al. 2002], JADE has two levels of abstraction where agents can be placed: the platform itself and containers, which also function as agent groups. This is a specificity of the JADE platform, which makes it necessary to evaluate the communication performance of agents (i) within the same container and (ii) in distinct containers.

For intra-container communication, the design of the experiment follows the same one as illustrated by ”Benchmark 4: IntraPlatform and IntraContainer”, in Figure 3. For communication between agents in different containers, the design of the experiment is illustrated by ”Benchmark 4: InterContainer”, also in Figure 3.

4.2. Scale-Up

To assess the platform’s scalability by employing the scale-up approach, we analyze how the variance in physical components such as CPU and Memory affect the average RTT performance. The chosen method in this experiment follows the design proposed in 4.1.1. This benchmark simulates a common behavior where each host has agents that behave actively and passively when it comes to initiating communication.

4.2.1. Scale-Up 1: CPU limit

For this assessment, we vary the number of CPU units that each JADE platform will have available, starting from 128 CPU units up to 2048 CPU units, while keeping the available memory at 2048MB.

4.2.2. Scale-Up 2: Memory limit

For this assessment, we vary the memory that each JADE platform will have available, starting from 128MB and going up to 2048MB. Similarly to the previous assessment, we maintain the CPU units available at 2048 units.

5. Results

5.1. Scale-Out Boxplot Results

5.1.1. Benchmark 1: Number of Hosts

For the first benchmark, it was possible to see a positive correlation when the number of hosts and the number of messages sent were increased. These results can be seen in Figure 4, but it shows no correlation to situations when the message size was increased.

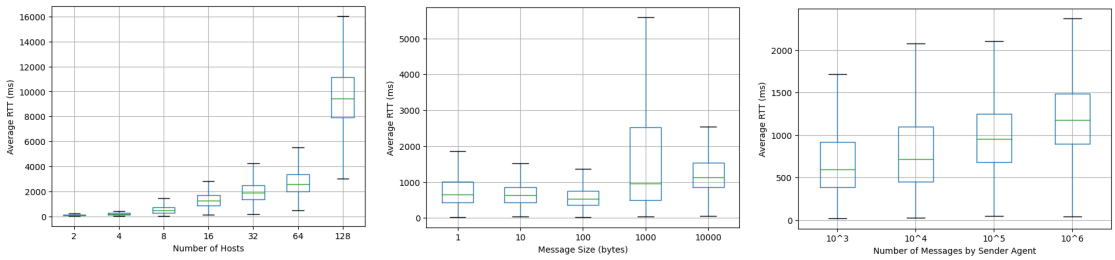

Figure 4. Scale-Out Benchmark 1 Results

5.1.2. Benchmark 2: Massive Reception on Single Receiver

For the second benchmark, it was not possible to observe any correlation between the increase of hosts and the increase of messages sent (Figure 5).

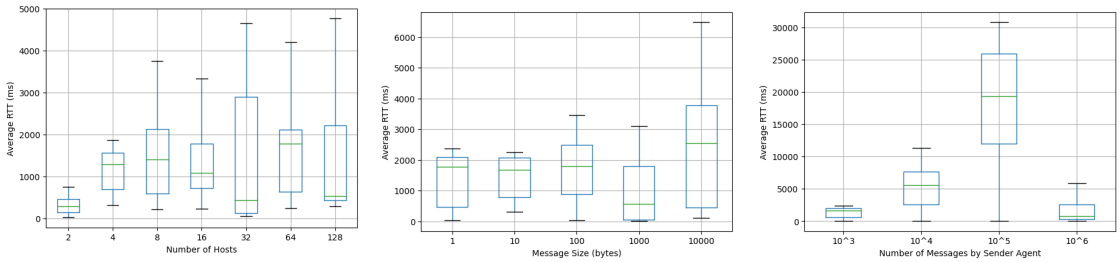

Figure 5. Scale-Out Benchmark 2 Results

5.1.3. Benchmark 3: Massive Reception on Multiple Receivers

This benchmark has shown the same pattern as seen in the 4.1.1. Increasing both the number of hosts in the system and the number of messages have a correlated impact on the average RTT, as shown in Figure 6. However, it was not possible to observe this pattern when varying the size of the messages.

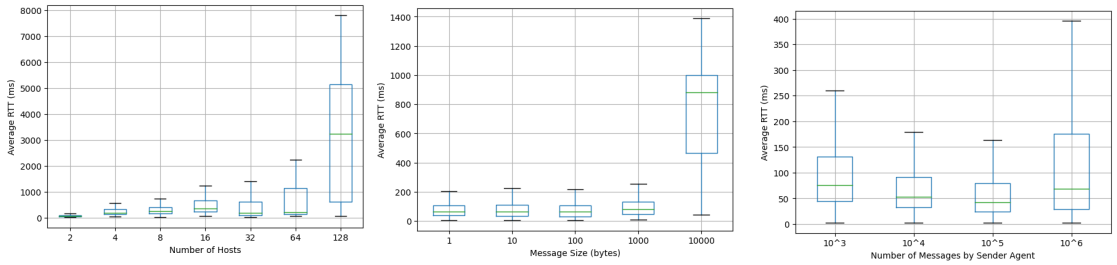

Figure 6. Scale-Out Benchmark 3 Results

5.1.4. Benchmark 4: Number of Agents per Host

For inter-platform communication, increasing the number of hosts and the number of messages sent also increased the average RTT, as can be seen in Figure 7. However, increasing the message size didn't provide any pattern that could be used to identify a relationship in the average RTT.

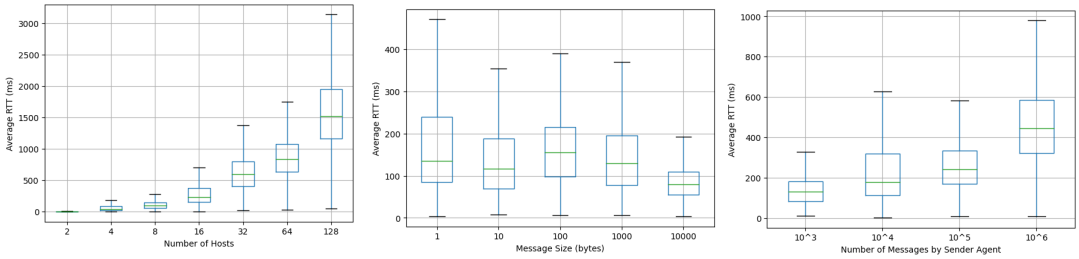

Figure 7. Scale-Out Benchmark 4 Results

When looking into intra-platform communication, the results differ. Changing the message size and the number of messages sent didn't result in a proportional variation in the average RTT (Figure 8), but increasing the number of hosts affected the metrics.

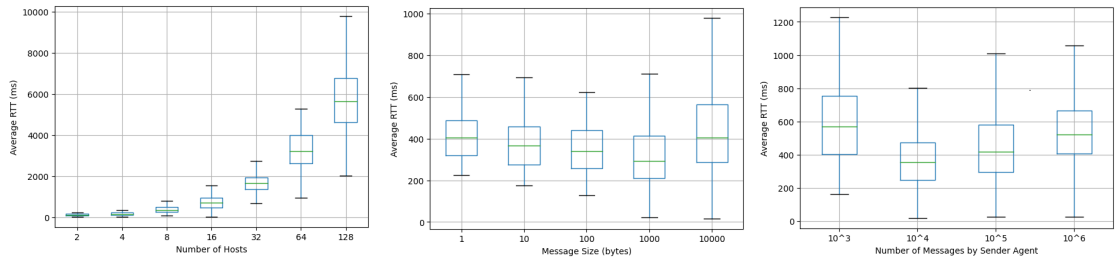

Figure 8. Scale-Out Benchmark 5 Results

The results for the experiments that stressed communication inter-container and intra-container presented a similar result. The correlation and possible limitation of the platform was presented when increasing (i) the number of agents in the experiment and (ii) the number of messages, as can be seen in Figures 9 (intra-container communication) and 10 (inter-container communication).

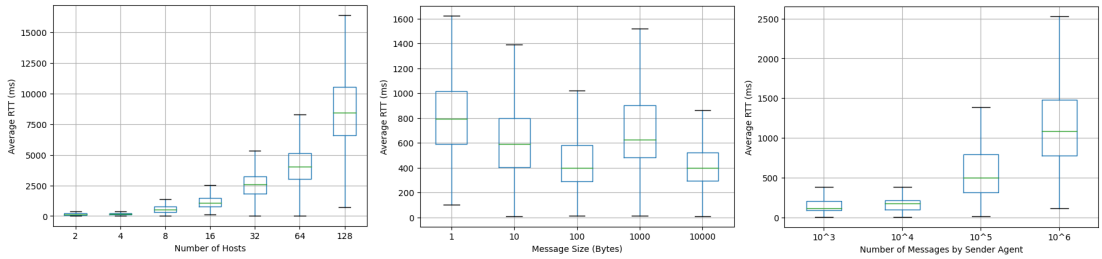

Figure 9. Scale-Out Benchmark 6 Results

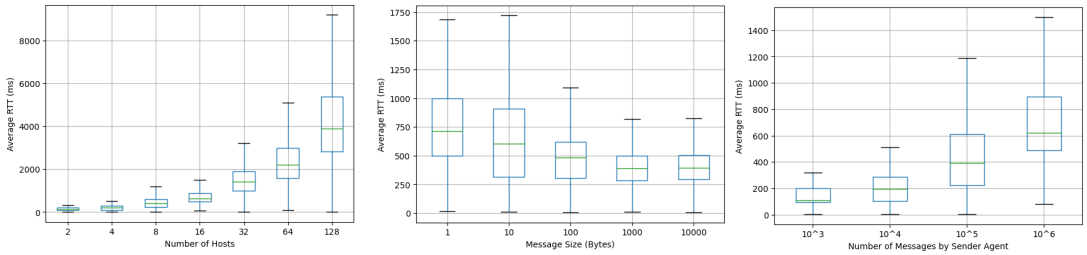

Figure 10. Scale-Out Benchmark 7 Results

5.2. T-test Variation Across Scale-Out Benchmarks

5.2.1. Number of Host Variation

The t-test results for the number of hosts variation indicate that all benchmarks have significant performance differences according to the increase in the number of hosts. The differences in the performance are substantial due to the large absolutes values observed in the t-statistics values, as can be seen in Table 1.

Table 1. Test T RTT Variation for Number of Hosts

Benchmark	Test T per Number of Hosts											
	2 - 4		4 - 8		8 - 16		16 - 32		32 - 64		64 - 128	
	Statistic	pValue	Statistic	pValue	Statistic	pValue	Statistic	pValue	Statistic	pValue	Statistic	pValue
Benchmark 1	-8.08	<0,05	-22.47	<0,05	-44.17	<0,05	-32.10	<0,05	-39.88	<0,05	-513.09	<0,05
Benchmark 2	-20.33	<0,05	-2.64	<0,05	-0.13	<0,05	- 0.003	<0,05	-1.34	<0,05	0.45	<0,05
Benchmark 3	-15.79	<0,05	-4.15	<0,05	-4.67	<0,05	0.70	<0,05	-1.64	<0,05	-10.11	<0,05
Benchmark 4	-69.35	<0,05	-44.66	<0,05	-96.90	<0,05	-151.75	<0,05	-112.40	<0,05	-259.08	<0,05
Benchmark 5	-4.23	<0,05	-9.17	<0,05	-6.61	<0,05	-18.18	<0,05	-25.50	<0,05	-20.84	<0,05
Benchmark 6	-9.09	<0,05	-93.56	<0,05	-92.22	<0,05	-164.05	<0,05	-149.99	<0,05	-394.66	<0,05
Benchmark 7	-17.88	<0,05	-52.85	<0,05	-54.35	<0,05	-129.50	<0,05	-123.44	<0,05	-634.65	<0,05

5.2.2. Message Size Variation

For the message size variation, it was not possible to identify an explicit correlation between the increase in the message size and the increase in the average RTT. When analyzing the results for the T-Test, it's possible to identify a variation for most of the cases, but the variation is not linear as can be seen in Table 2.

Table 2. Test T RTT Variation for Message Size

Test T per Message Size (bytes)								
Benchmark	1 - 10		10 - 100		100 - 1000		1000 - 10000	
	Statistic	pValue	Statistic	pValue	Statistic	pValue	Statistic	pValue
Benchmark 1	21.86	<0.05	34.70	<0.05	-86.47	<0.05	53.87	<0.05
Benchmark 2	-0.31	0.75	-1.94	0.05	6.52	<0.05	-12.37	<0.05
Benchmark 3	-2.16	<0.05	3.33	<0.05	-8.02	<0.05	-28.88	<0.05
Benchmark 4	23.76	<0.05	-24.47	<0.05	17.84	<0.05	59.46	<0.05
Benchmark 5	1.55	0.12	1.13	0.25	1.78	0.07	-10.60	<0.05
Benchmark 6	31.48	<0.05	43.89	<0.05	-58.98	<0.05	67.63	<0.05
Benchmark 7	18.23	<0.05	33.87	<0.05	33.01	<0.05	-4.43	<0.05

5.2.3. Number of Message Variation

When increasing the number of message exchanges between agents, the t-test results for the number of messages data indicate that all benchmarks have significant performance differences across all number of message ranges, as can be seen in Table 3. For most of the benchmarks, when varying the number of messages, the t-statistics values are negative, indicating that the first group (with lower messages being sent) has a lower mean than the second group (with a higher number of messages sent). This is concluded by the pValue obtained being lower than 0.05.

Table 3. Test T RTT Variation for Number Of Messages

Test T per Number of Message						
Benchmark	10^3 - 10^4		10^4 - 10^5		10^5 - 10^6	
	Statistic	pValue	Statistic	pValue	Statistic	pValue
Benchmark 1	-41.16	<0.05	-96.47	<0.05	-175.47	<0.05
Benchmark 2	-33.52	<0.05	-74.99	<0.05	86.06	<0.05
Benchmark 3	6.69	<0.05	9.19	<0.05	-129.65	<0.05
Benchmark 4	-75.86	<0.05	-50.37	<0.05	-1271.23	<0.05
Benchmark 5	8.45	<0.05	-17.00	<0.05	-34.75	<0.05
Benchmark 6	-43.81	<0.05	-370.19	<0.05	-1280.49	<0.05
Benchmark 7	-36.32	<0.05	-146.96	<0.05	-1405.03	<0.05

6. Scale-Out Boxplot Results

When limiting each host to use CPU units from 128 to 2058, it was evidenced that the performance of the platform was affected (Figure 11).

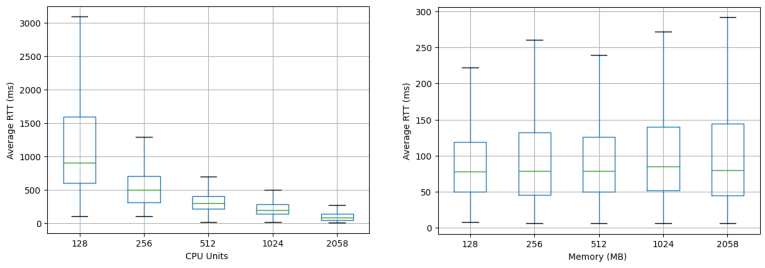

Figure 11. Scale-Up Benchmark Results

Differently from the variation of the CPU Units, the memory limit didn't show

any correlation between the average RTT and the available memory when the host had at least 128MB to operate, as shown in Figure 11.

6.1. T-Test Variation Across Scale-Up Benchmarks

In summary, the t-test results for the resource allocation showed that the increase of CPU leverage led to a reduction of the average RTT mean in all scenarios, while the variation of memory provided no direction correlation between a higher resource allocation and an impact in the average RTT.

Table 4. Test T RTT Variation for Message Size

Test T per Resource								
Benchmark	128 - 256		256 - 512		512 - 1024		1024 - 2048	
	Statistic	pValue	Statistic	pValue	Statistic	pValue	Statistic	pValue
Benchmark 1	65.70	<0.05	40.57	<0.05	28.35	<0.05	58.63	<0.05
Benchmark 2	-13.55	0.75	2.65	0.05	-17.27	<0.05	-7.63	<0.05

7. Discussion and future work

This experiment analyzed the response of the JADE platform in specific contexts for scale-up and scale-out scalability aspects. The performance evaluation approach presented in the current work was first adopted by [Rodrigues 2019], and [Alencar 2020] continued these experiments by deploying the agents into a cloud-based infrastructure. Our work analyzes the response of the JADE platform within a larger system with hundreds of platforms, and it also includes the analysis of the platform from a scale-up approach.

When analyzing the results from the scale-out experiments, it's possible to see that the number of hosts was a common bottleneck factor for most of the experiments, followed by the number of messages sent by each agent. This is an indicator that for a large distributed system, the HTTP MTP protocol might be a limiting factor for the overall system's scalability. In the scale-up approach, varying the memory available for the hosts didn't directly interfere with the average RTT, but the average RTT decreased with the increase of the number of CPU Units available per host, thus indicating that for scaling large distributed MAS that require fast response, making more CPU units available should be the priority.

In the future, the present work can be replicated for other MAS platforms in order to understand how they behave according to the benchmarks used, which can provide an important indicator of whether or not a platform might be used when considered for large-scale MAS systems that need to adhere to specific performance conditions or that must be deployed within devices with hardware limitations. All tables and figures, as well as the data used for analyzing the experiments, are available online [3].

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001

[3]https://github.com/lti-usp/2023-wesaac-sgursky-casals-brandao/

References

Alencar, R. F. (2020). Escalabilidade e comunicação. Undergraduate thesis, Escola Politécnica da Universidade de São Paulo, São Paulo.

Balke, T. and Gilbert, N. (2014). How do agents make decisions? a survey. *JASSS*, 17:13.

Bellifemine, F., Poggi, A., and Rimassa, G. (1999). *JADE - A FIPA-compliant agent framework*, pages 97–108. The Practical Application Company Ltd.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Developing multi-agent systems with a fipa-compliant agent framework. *Softw., Pract. Exper.*, 31:103–128.

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. John Wiley & Sons.

Ferber, J. (1999). *Multi-agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley.

Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296.

Poslad, S. and Charlton, P. (2001). *Standardizing Agent Interoperability: The FIPA Approach*, volume 2086, pages 98–117. Springer.

Rodrigues, H. (2019). Avaliação de escalabilidade e desempenho da camada de transporte de mensagens em plataformas multiagente. Master's thesis, Escola Politécnica da Universidade de São Paulo.

Russell, S. and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*. Pearson, New York, NY, 2nd edition.

Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8:345–383.

Such, J., Alberola, J., Mulet, L., Espinosa, A., García-Fornes, A., and Botti, V. (2007). Large-scale multiagent platform benchmarks. *Proceedings of the Multi-Agent Logics, Languages, and Organisations-Federated Workshops (LADS07)*.

Tanenbaum, A. S. (2003). *Computer networks*. Pearson Education India.

Thangarajah, J., Padgham, L., and Harland, J. (2002). Representation and reasoning for goals in bdi agents. *Australian Computer Science Communications*, 24(1):259–265.

Vitaglione, G., Quarta, F., and Cortese, E. (2002). Scalability and performance of jade message transport system. *Autonomous Agents and Multi-Agent Systems*.