

SURVEY PAPER

Survey on the run-time systems of enterprise application integration platforms focusing on performance

Daniela L. Freire | Rafael Z. Frantz  | Fabricia Roos-Frantz | Sandro Sawicki

Department of Exact Sciences and Engineering, Unijuí University, Ijuí-RS, Brazil

Correspondence

Rafael Z. Frantz, Department of Exact Sciences and Engineering, Unijuí University, Ijuí-RS 98700-000, Brazil.
Email: rzfrantz@unijui.edu.br

Funding information

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Grant/Award Number: 73318345415 and 88881.119518/2016-01; Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul, Grant/Award Number: 17/2551-0001206-2

Summary

Companies are taking advantage of cloud computing to upgrade their business processes. Cloud computing requires interaction with many kinds of applications, so it is necessary to improve the performance of software tools that allow keeping information on all these applications consistent and synchronised. Integration platforms are specialised software tools that provide support to design, implement, run, and monitor integration solutions, which aim to orchestrate a set of applications so as to promote compatibility among their data or to develop new functionality on top of the current ones. The run-time system is the part of the integration platform responsible for running the integration solutions, which makes its performance the uttermost important issue. The contribution of this article is two-fold: a framework and an evaluation of integration platforms. The former is a framework composed of ten properties grouped into two dimensions to evaluate the run-time systems focusing on performance. Using this framework as reference, the second contribution is an evaluation of nine open-source integration platforms, which represent the state-of-the-art, provide support to the integration patterns, and follow the pipes-and-filters architectural style. In addition, as a result of this work, we suggest open research directions that can be explored to improve the performance of the run-time systems and at the same time may be useful to adapt them to the context of cloud computing.

KEYWORDS

cloud computing, enterprise application integration, integration patterns, integration platform, run-time system

1 | INTRODUCTION

Companies often need to use their software ecosystems^{1,2} to support and improve their business processes. These ecosystems are composed of many applications, usually designed without taking into account their possible integration. The field of study known as Enterprise Integration Applications (EAI) seeks to provide methodologies, techniques, and tools for the design and implementation of integration solutions.³ In general terms, an integration solution aims to orchestrate a set of applications to keep them synchronised or provide new features that can be built from those already developed. An integration solution is composed of processes that contain the integration logic and ports that encapsulate adapters with communication protocols to connect applications to the integration solution.

Integration platforms are specialised software tools that provide support to design, implement, run, and monitor integration solutions. In the last years, many message-based integration platforms have been created by the EAI community. These platforms have been heavily influenced by the catalogue of conceptual integration patterns documented by Hohpe and Woolf⁴ and have followed the architectural style of pipes-and-filters.⁵ In an integration solution, pipes represent message channels, and filters represent atomic tasks that implement a concrete integration pattern to process encapsulated data in messages. The adoption of this architecture allows for the desynchronisation of tasks that make up the integration solution.

There are several open-source message-based integration platforms that can be used to build integration solutions, such as Mule,⁶ Apache Camel,⁷ Spring Integration,⁸ Fuse,⁹ ServiceMix,¹⁰ Petals,¹¹ Jitterbit,¹² WSO2 ESB,¹³ and Guaraná.¹⁴ A software is labelled open source when its source code is made available under a licence that grants users the rights to study, change, and distribute the software for any purpose.¹⁵ Usually, these integration platforms provide a domain-specific language, a development toolkit, a testing environment, a monitoring tool, and a run-time system. The domain-specific language is focused on the elaboration of conceptual models for the integration solution, with a level of abstraction close to the domain of the problem. The development toolkit is a set of tools that allows the implementation of the solution, that is, the transformation of the conceptual model into executable code. The testing environment allows testing individual parts or the entire integration solution, with the aim of identifying and eliminating possible defects in the implementation. The monitoring tool is used to monitor, at run time, the operation of the integration solution and detects errors that may occur during message processing. The run-time system provides all the support required to execute these integration solutions.

Cloud Computing¹⁶ is another field of research that has drawn the attention of the scientific community and represents a new paradigm of development, commercialisation, and use of software. This field has been transforming the current software ecosystems and revolutionising the way companies provide computer support to their business processes. Cloud computing enables companies to hire service packages by dramatically reducing their total cost of ownership with the information technology (IT) infrastructure, without sacrificing the quality of the IT support provided.^{17,18} This is due mainly to the pay-as-you-go charging model that allows users to pay for cloud computing based on the amount of computing resources consumed.¹⁹ Along with the pay-as-you-go model, cloud computing has also brought the elasticity feature, which allows increasing and decreasing computational resources to better meet the demands of applications running in the cloud infrastructure.²⁰ The number of independent processing units (known as cores) and the amount of memory are central types of computational resources in the cloud involved in the process of deploying applications to the cloud. The number of cores directly influence the execution of threads created in computer applications and, thus, their performance.²¹ The advancement of cloud computing technologies has led companies to a major transformation in their software ecosystem, which now includes on-premise applications, migrated applications to virtual machines in the cloud, mobile applications, social media applications, and many other software available in the cloud as services.²²

The quality of service that integration solutions are able to achieve in terms of message processing is directly related to the run-time system of the integration platform.²³ In this context, a run-time system performs better when an integration solution is capable of processing more messages per unit of time. Typically, in order to achieve the desired quality of service with an integration solution, software engineers have increased computational resources in the server machine in which the integration platform is installed within the enterprise. This approach links the increased efficiency of execution of an integration solution to the increase in financial costs required to augment the current hardware or the purchase of a new server with greater processing power that can generate the desired impact on the performance of the run-time systems, thus increasing the number of messages processed by the integration solutions.

The hiring of virtual machines in the cloud to host integration platforms allows a reduction of the total cost of ownership for the realisation of EAI by the companies, as well as by means of the feature of elasticity of the cloud, a greater flexibility for the increment of computational resources.²⁴⁻²⁶ This fact has motivated many integration platform providers to migrate and offer a cloud version of their platforms to run integration solutions in the cloud.¹⁸ The migration of integration platforms to virtual machines in the cloud has given rise to a new service model that is being referred to by the EAI community as integration Platform-as-a-Service (iPaaS).²⁷ Figure 1 presents a conceptual map in which we abstract the key concepts involved in the research context of this article.

Data from 2015 shows that, together, Latin America, Central America, and North America account for 67% of the market for iPaaS integration platforms, followed by Europe, the Middle East, and Africa, which together account for 22%, and Asia and the Pacific with 11% of this market, these values should remain, with little oscillation, until the end of 2019.²⁸ The traditional market for integration platforms used on-premise registered growth of less than 10% in 2016, whereas the market for iPaaS integration platforms had a 60% expansion over the previous year, representing a global market of 700 million dollars.²⁹ As early as 2017, two out of three application integration projects have been developed directly with

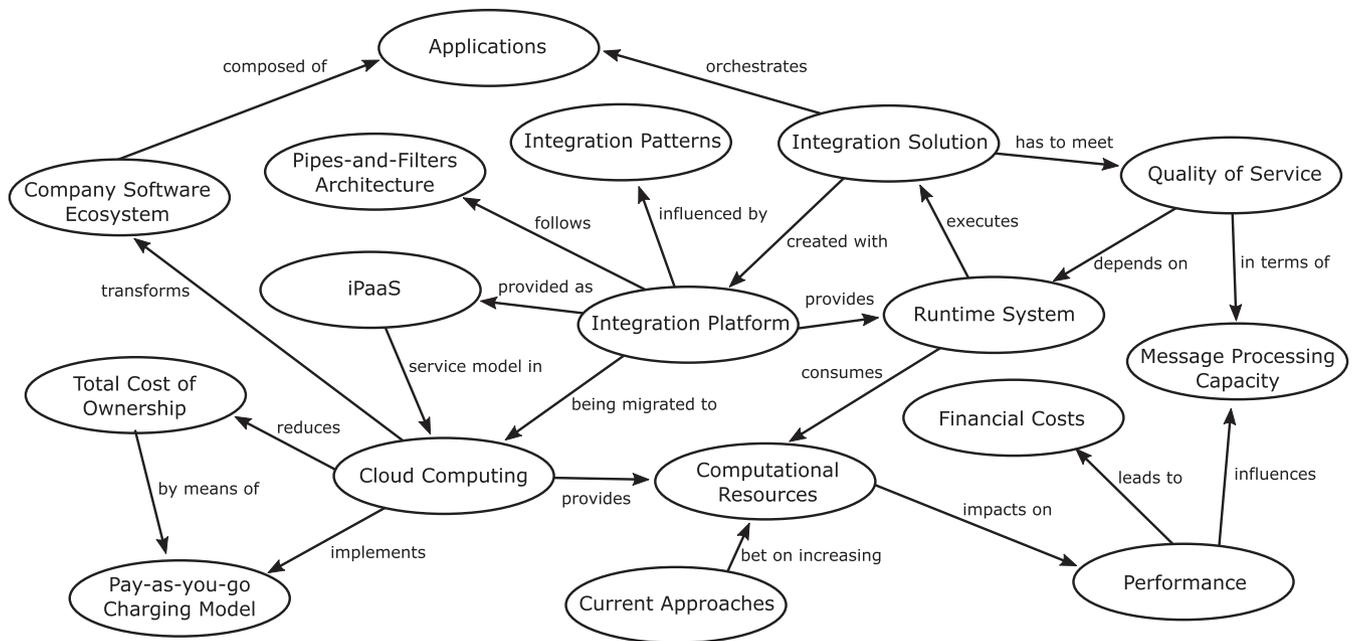


FIGURE 1 Conceptual map of the key concepts in the research context. iPaaS, Platform-as-a-Service

cloud integration platforms.³⁰ The investment made by companies on iPaaS integration platforms will increase by 40% by 2019,²⁸ making iPaaS the preferred integration platform by companies, with annual revenue growth higher than the traditional platform market of integration used on-premise.^{29,31}

Given the high investments on iPaaS, a current challenge for the providers of integration platforms in the cloud is to adapt their platforms to ensure they are suitable for cloud deployment, as well as reducing costs by optimising resource usage.^{32,33} In this context, the efficiency of run-time systems is fundamental since the cloud follows the pay-as-you-go model, and therefore, there is a direct impact on the financial cost involved in executing integration solutions. The higher the efficiency of run-time systems, the less computational resources will need to be hired or consumed for an integration solution to increase the efficiency of its execution and achieve the expected quality of service. In this article, we introduce a comparison framework to evaluate integration platforms regarding performance of their run-time systems. This framework is composed of ten performance properties organised into two dimensions. We analyse and compare nine integration platforms and suggest open research directions that can be explored to improve the performance of the run-time systems and at the same time may be useful to adapt them to the context of cloud computing.

The rest of this article is organised as follows: Section 2 presents the research method we follow in this survey; Section 3 discusses related work; Section 4 describes the dimensions and the respective properties that make up the evaluation framework; Section 5 describes how we selected the open-source message-based integration platforms; Section 6 analyses the integration platforms following the evaluate framework; Section 7 reports on the issues to be investigated derived from the survey; Section 8 discusses the threats to validity in this research; and Section 9 presents our conclusions.

2 | RESEARCH METHOD

In this section, we present the research method we follow in this article to review integration platforms and find out research directions in the context of performance of their run-time systems. We construct a comparison framework made up of performance properties and apply this framework to nine integration platforms. Our research method is abstracted in Figure 2. It has two main activities: framework construction and framework application. In the former, we conducted a feature analysis by means of a qualitative screening research on academic literature³⁴ and on technical literature³⁵ to identify performance properties that can be used by academy and industry. In the latter, the comparison framework was applied to nine integration platforms and resulted in six research directions. Screening research is one of the forms of feature analysis, in which the evaluation is performed by software engineers based on documentation only. Screening is

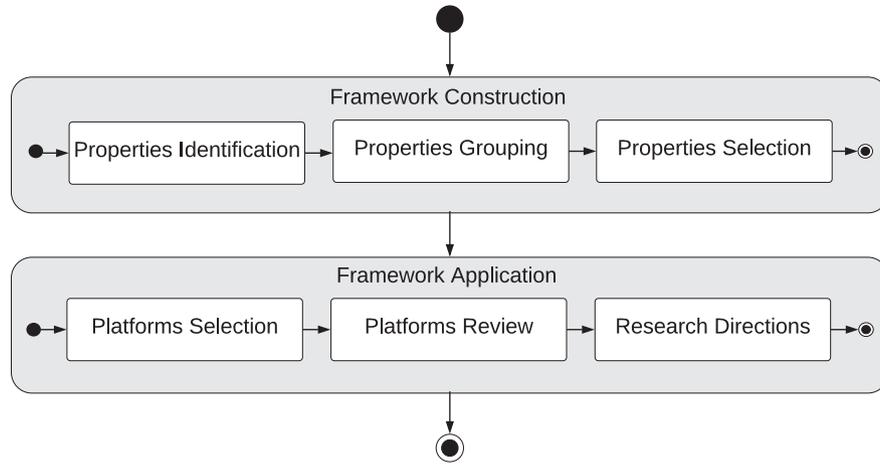


FIGURE 2 Research method applied

indicated for a more complex evaluation, in which it is possible to reduce a large number of platforms to a short list that can be deeply evaluated.³⁶

Feature analysis is constructed from the selected research papers in multivocal literature review (MLR). Ogawa and Malen³⁷ states that “multivocal literature is comprised of all accessible writings on a common, often contemporary topic. The writings embody the views or voices of diverse sets of authors (academics, practitioners, journalists, policy centres, state offices of education, local school districts, independent research and development firms, and others). The writings appear in a variety of forms. They reflect different purposes, perspectives, and information bases. They address different aspects of the topic and incorporate different research or nonresearch logics.” MLRs recently started to be used in software engineering; thus, we conducted and reported our study based in the guidelines documented by Garousi et al³⁵ to ensure high quality of our MLR processes and their results.

2.1 | Framework construction

In this activity, we produced a comparison framework to analyse the run-time systems of the integration platforms. This framework provides properties that contribute to endow current run-time systems of integration platforms with quality attributes to increase their performance and at the same time can lead to efficiency in the execution of integration solutions. This activity is based on the coding process,³⁸ by which data is broken down, conceptualised, and synthesised. The main coding procedures are the following:

- Properties Identification - aims at identifying quality attributes for run-time systems that allow to compare and group similar performance properties. We constructed an initial list of properties that have an impacted on the processing time of messages in the integration solutions and in the consumption of computational resources of the run-time system.
- Properties Grouping - aims at identifying which problems in the execution of integration solutions are solved when the run-time system are endowed with the identified properties. We classified the properties into dimensions and tabulated the relationships between properties and dimensions. This tabulation allowed us to identify redundancies among the dimensions.
- Properties Selection - aims at evaluating the dimensions and properties to select a consistent subset of them. This selection was carry out based on our experience of several years on the development of integration projects in real-world software ecosystems. The subset of dimensions and their properties is the comparison framework used as criteria to evaluate integration platforms, in relation to the performance of their run-time systems.

This activity resulted in a set of ten performance properties is grouped in two dimensions: message processing and fairness execution. The former dimension relates to improving the efficiency of processing a message by run-time system, which can also be seen as increasing the average number of messages processed per unit of time. The latter dimension is concerned with the assignment of threads to tasks aiming at a fair execution to help to minimise the average time that a message takes to be processed in the integration solution.

Once the coding derived quality attributes, it is required to contextualise the platforms in comparison to each other. In other words, we aim at evaluating the quality performance features from integration platforms considering the following context: open source, provide support to the integration patterns,⁴ and follow the pipes-and-filters architectural style.⁵ Thus, we aim at answering the following research questions:

- RQ1: What are the relevant features that can help to improve the performance of run-time systems of the integration platforms?
- RQ2: What are the state-of-the-art integration platforms that provide support to integration patterns and follow the pipes-and-filters architectural style?
- RQ3: Are the state-of-the-art open-source message-based integration platforms endowed with features that can help to improve the performance of their run-time systems from the perspective of message processing and fairness execution to the context of cloud computing?

Our hypothesis is that the run-time systems are not sufficiently endowed with features to improve message processing and to promote a fair execution of tasks in the context of cloud computing. Based on this survey, we suggest open research directions that can be explored in order to improve the performance of the run-time systems and at the same time may be useful to adapt them to the context of cloud computing.

2.2 | Framework application

In this step, we applied the comparison framework for nine the integration platforms. The evaluation of the platforms was carried out based on a deep study of them and in our knowledge of their use in integration projects in real-world software ecosystems. The main procedures are listed below and detailed in next sections:

- Platforms Selection - aims to select integration platforms, in which it is possible to compare their quality attributes regarding performance. We collected 42 platforms and formed an initial list. After, we applied inclusion criteria to filter and homogenise the set of selected platforms, remained nine platforms.
- Platforms Review - aims at identifying values for performance properties of the comparison framework. We carried out an MLR on the publicly available source code and the documentation from their web site, books, and academic articles in order to study and evaluate integration platforms.
- Research Directions - aims at identifying the lack of features that can to increase the performance of run time and at the same time can lead to efficiency in the execution of integration solutions. Based on this survey, we suggest open research directions that can be explored in order to improve the integration platforms and at the same time may be useful to adapt them to the context of cloud computing.

3 | RELATED WORK

In our review of the literature, we identified proposals that analyse integration platforms by means of high level of abstraction properties, whereas we analyse run-time systems of integration platforms focusing on their performance. Corchuelo et al³⁹ analysed integration platforms taking into account the following groups of properties: scope of the tool, modelling capabilities, and technical features. Scope of the tool includes essential properties that an integration platform must have, as their lack can be prohibitive. Modelling capabilities deal with important but not essential properties, that is, the absence of such properties makes integration modelling more complex and less intuitive. Technical features address properties that affect the ease of programming or management of integration solutions. In their work, Corchuelo et al analysed five integration platforms, namely, Camel, Mule, ServiceMix, Spring Integration, and BizTalk. Their work focuses on general-purpose properties of the integration platforms, whereas we focus exclusively on performance properties. García-Jiménez et al⁴⁰ provided an evaluation based on features and timing analysis. The evaluated features were the support for open standards, the documentation and implementation support, and the usability of the platform regarding functionality and its graphical user interface. Regarding the timing analysis, their work is based on an existing system aiming at obtaining useful scenarios and measures for being considered in business integration. In their work, they compared the following open-source Enterprise Service Buses (ESB): Fuse, Mule, and Petals. In their work, Garcia et al made an evaluation by means of experiments, measuring the response time in the invocations. The proposal of these authors is different from ours, in the sense that we aim to evaluate run-time systems by means of the study of ten different

TABLE 1 Properties approached in the related works

Related Work	Patterns Support	Usability	Management	Maintainability	Performance
Corchuelo et al ³⁹	✓	✓	✓		
García-Jiménez et al ⁴⁰	✓	✓		✓	
Palanimalai and Paramasivam ⁴¹		✓	✓		
Bhadoria et al ⁴²		✓	✓	✓	✓ [†]
Ritter et al ⁴³	✓				
Our proposal					✓

[†]Approaches performance in general by measuring a single property.

performance properties. Palanimalai and Paramasivam⁴¹ presented issues of security, scalability, elasticity, monitoring, and management, as cloud integration challenges and key aspects to choose an integration platform. They propose an hybrid integration architecture, a model for integration of on-premise applications with cloud-based applications, whose capabilities should include seamless integration, complete platform extension, elasticity, usability, and agility. Palanimalai and Paramasivam argue that this model delivers a secure gateway for data secured communications between cloud-based integration and services in an ESB. While they propose an integration architecture that addresses generic capabilities, our work address specifically capabilities regarding the performance of the run-time systems. Bhadoria et al⁴² produced an analytic survey of ESB. In their work, they compared 12 ESBs in the market, classifying if they are open source, and which ones provide a graphical interface to its clients. Bhadoria et al analyse aspects of performance, flexibility, and service governance. They compare them on the base of security metric, deployment scenario, virtual environment support, and essential features like message routing, transformation, and protocol conversion. Their work deals with performance in general only as one of the features, whereas our work deals with the performance as the central subject. Ritter et al⁴³ made a systematic literature review of cloud and hybrid (on-premise with the cloud) application integration proposals and analysed fifteen integration platforms to evaluate the support for integration patterns originally documented by Hohpe and Woolf⁴ provided to scenarios of cloud integration. Ritter et al also made a quantitative analysis of real-world integration scenarios and concluded that the original integration patterns do not cover new trends and nonfunctional aspects for enterprise application integration, such as security, monitoring, and storage. Their work aims at filling the gaps in existing patterns for new integration trends and for some nonfunctional aspects, whereas our work aims to evaluate exclusively the performance of nonfunctional aspects of run-time systems. The related work is summarised in Table 1, by indicating the properties approached in each one of them.

4 | EVALUATION FRAMEWORK

In this section, we present a set of properties that guided our analysis of the integration platforms to answer the first research question (RQ1). These properties are grouped into two dimensions, namely, message processing and fairness execution. These dimensions can contribute to current run-time systems by endowing integration platforms with performance features and, at the same time, lead to an efficient execution of integration solutions. The following sections detail the respective properties in each dimension.

4.1 | Message processing

This dimension addresses the improvement of the efficiency of the run-time system to process a message, which can also be seen as increasing the average number of messages processed per unit of time. The properties of the message processing dimension are related to the capacity of reducing the demanded real-time to completely process a message by an integration solution. These properties are described as follows:

- **Designed for multicore.** This property indicates whether the run-time system has been developed to take advantage of multicore or not. Multicore programming has to be carried out in order to take full advantage of multiple cores available in the hardware processors. Nowadays, this becomes an increasingly important requirement in the use of the powerful many-core parallel machines that compose the computing infrastructures.⁴⁴ Multiple cores work together to increase the capability of processing multiple tasks or to increase the performance of the system by operating on

multiple instructions simultaneously in an efficient manner. This property may take the following values: *yes* or *no*. *yes* indicates that the run-time system was developed to take advantage of multicore hardware; otherwise, the value is *no*.

- **Thread pool configuration.** This property indicates how threads are managed in thread pools. The programming languages, in which the run-time systems are developed, have objects that encapsulate functions to create and manage threads with predefined settings. Furthermore, they also have utility methods that allow a custom configuration to provide more flexibility, such as setting the maximum number of threads allowed in a thread pool, the maximum time a thread remains idle, and the type of queue used for the tasks waiting to be executed. This property may take the following values: *fixed*, *limited*, or *elastic*. *fixed* indicates that a thread pool is composed of a fixed number of threads, which is known at design time; *limited* indicates the number of threads in a thread pool can increase automatically during run time until a threshold defined at design time is reached; *elastic* indicates that the number of threads in a pool can automatically increase and decrease during run time within a range of values defined at design time.⁴⁵
- **Type of message storage in process.** This property indicates how the run-time system deals with the storing of messages during the execution of an integration solution. Storing messages in-memory is faster but can be more expensive.⁴⁶ Messages that contain a big amount of data impact the amount of memory required for their processing inside the integration solutions. In such cases, rather than storing messages only in-memory, the run-time system can store them in-disc. This property may take the following values: *memory* or *hybrid*. *memory* indicates that the run-time system stores messages only in-memory. *hybrid* indicates that the run-time system adopts different strategies for storing messages, such as combining in-disc and in-memory.
- **Distributed process execution.** This property indicates if an integration solution can be divided and distributed to different machines to execute a set of correlated messages, thereby promoting scalability.⁴⁷ This property allows increasing the number of tasks executed per unit of time in cases in which there is no dependency between the tasks, and the input data of one task are the output data produced by another task. This property may take the following values: *yes* or *no*. *yes* indicates that the run-time system takes advantage of scalability; otherwise, the value is *no*.
- **Thread pool creation.** This property indicates the way and the stage at which thread pools can be created. Historical and current execution data can be used by run-time systems to make decisions during run time.⁴⁸ Such data can point to optimised strategies for the creation of threads, empowering run-time systems to deal with different workloads of messages. This property may take the following values: *dynamic* or *static*. *dynamic* indicates that thread pool creation is done at run time by means of information taken from the run-time system. *static* indicates that the thread pool is created at design time by software engineers.

4.2 | Fairness execution

This dimension addresses the assignment of threads to tasks, in a balanced way, to minimise the average time that a message takes to be processed in an integration solution. The following properties provide means that contribute to have a fair execution of tasks:

- **Starvation Detection.** This property indicates if the run-time system is endowed with the capacity to detect tasks that do not execute within an accepted time frame. Roughly speaking, starvation happens when a thread cannot fetch tasks because other threads have effectively blocked it from doing so.⁴⁹ This property may take the following values: *yes* or *no*. *yes* indicates that the run-time system is endowed with intelligence to detect hot spots during the execution of the integration solutions; otherwise, the value is *no*.
- **Thread Pool Policy.** This property indicates the policy followed by the run-time system to schedule the execution of tasks of an integration solution to computational resources. In cloud environments, scheduling of an integration solution becomes challenging because its performance must result in reduced scheduling overhead, minimised cost, and maximise resource utilisation while still meeting the specified deadline.⁵⁰ However, cloud environments usually cause computing overheads that negatively impacts on the overall performance and costs of the workflow execution.⁵¹ This property may take the following values: *fifo*, *priority*, or *mapping*. *fifo* means that the run-time system follows first-in–first-out policy, in which threads are assigned to tasks in order that they arrive; *priority* means that the run-time system allows tasks to have priority associated to them, so influencing the scheduling; and *mapping* means that the scheduling policy follows a mapping based on a mathematical model or optimisation method that allows finding an optimal scheduling policy by previously evaluating the integration solution.

- **Task Complexity.** This property indicates if the run-time system takes into consideration the computational complexity of tasks to assign threads. Tasks that perform more complex operations tend to be implemented with more computational instructions, therefore requires a longer time to be executed. Task granularity must be considered for reducing the impact of overheads on the execution of an integration solution in the cloud environment since it can lead to an inefficient resource utilisation, resulting in an unfavourable application throughput.⁵² This property may take the following values: *yes* or *no*. *yes* indicates that the run-time system recognises the computational complexity of each of the tasks and this can be utilised to decide the order of execution of tasks, so that tasks of less computational complexity can be executed first; if the run-time system does not recognise the computational complexity, the value is *no*.
- **Execution Model.** This property indicates the execution model implemented by the run-time system, which deals with the level of the execution of an integration solution. It is possible to classify these models in process-based and task-based. In the former, the run-time system controls process instances as a whole, ie, there are no means that it can interact with the internal tasks. In the latter, the run-time system may control both process instances and their internal tasks. The literature shows that the task-based model offers better performance with a steady stream of data input and lower performance when the input rate increases,⁵³ although this model is more complex to provide transaction and fault-tolerance support.¹⁴ This property may take the following values: *process-based*, *task-based*, or *hybrid*. *process-based* indicates that the run-time system adopts a process-based model; *task-based* indicates that the run-time system adopts a task-based model. *hybrid* model indicates that the run-time system will adopt the model which best fits the execution profile regarding predefined parameters, such as message input rate, number of processors, or average message size.
- **Throttling controller.** This property indicates if the run-time system allows to control the rate of incoming messages in an integration solution, so that when this rate exceeds a previously determined limit, the run-time system can adopt suitable policies to preserve the execution of the integration solution. Such intervention policies may be (i) refusing new messages or (ii) buffering at input the incoming messages or persisting them in a repository. This property may take the following values: *no* or *yes*. *yes* indicates that the run-time system can control the rate of incoming messages, that is, it has a throttling controller⁴; otherwise, the value is *no*.

5 | INTEGRATION PLATFORMS SELECTION

In this section, we present the research methodology that we applied to select the integration platforms analysed in this article. Motivated by the second research question (RQ2), we reviewed the literature to identify the state-of-the-art open-source message-based integration platforms and to conduct a study of their run-time systems. The research methodology is abstracted in Figure 3 and is composed of three steps: collection of references, collection of integration platforms, and selection of integration platforms. In the step for collection of references, we selected scientific articles and technical reports regarding integration platforms based on predefined search string. In the step for collection of integration platforms, we analysed these documents and extracted a list of integration platforms based on inclusion criteria “cited platforms explicitly.” In the step for selection of integration platforms, we selected integration platforms based on inclusion criteria “open source, provide support to the integration patterns,⁴ and follow the pipes-and-filters architectural style.⁵” The methodology and steps are explained in the following sections.

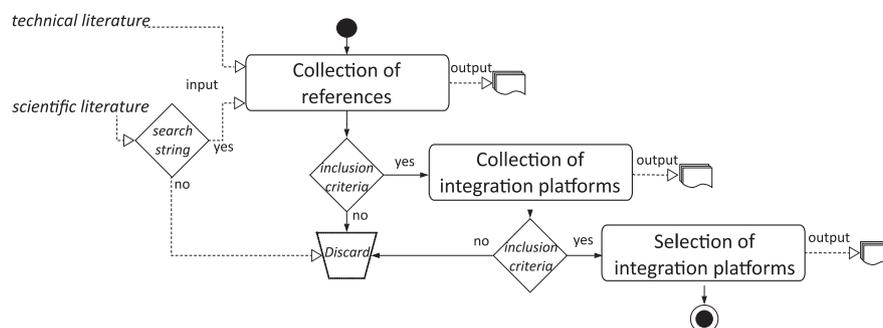


FIGURE 3 Integration platforms selection methodology

TABLE 2 Review of the state-of-art integration platforms

Literature	Year	Collected	Selected	References
<i>Scientific</i>	2013	29	3	54-56
	2014	20	2	57,58
	2015	31	4	41,59-61
	2016	38	5	3,62-65
	2017	30	4	42,66-68
	2018	10	1	69
<i>Technical</i>	2016	-	1	70
	2017	-	2	29,31

5.1 | Collection of references

In this step, we performed a search on Scopus using the following search string: (“enterprise application integration” or “integration systems” or “business integration”) and (“tool” or “platform”). This study searched for published articles from January 2013 to April 2018 (date that the research was done), written in English, in the subject area of Computer Science. The search returned 158 unique results, which cover a diverse range of journals and conferences (see Table 2).

Next, titles and abstracts of these 158 articles were carefully reviewed to select those articles that explicitly make reference to at least one integration platform. At the end, there were 19 articles. Additionally, we revised the technical literature by means of 3 reports regarding integration platforms: Gartner,^{29,71} Forrester,⁷⁰ and Ovum.³¹

5.2 | Collection of integration platforms

In this step, we collected 42 platforms from the articles and reports, cf Table 3. In this table, the first column identifies the platform; the second column indicates if the platform is released under an open-source licence; the third column indicates if the platform supports the enterprise integration patterns (EIPs)⁴; the fourth column indicates if the platform adopts the pipes-and-filters architectural style⁵; and the last column provides the references to the selected scientific articles and technical reports.

5.3 | Selection of integration platforms

In this step, we filtered the 42 integration platforms by considering only those that are released under an open-source licence, support the EIPs, and that adopt the pipes-and-filters architectural style. At the end, 9 integration platforms were selected to be analysed, cf Table 4. The chosen platforms were Camel,⁷ Guaraná,⁵³ Fuse,⁹ Jitterbit,¹² Mule,⁶ Petal,¹¹ ServiceMix,¹⁰ Spring Integration,⁸ and WSO2.¹³ The analyses of the platform was carried out considering books, online documentation, and source code accessible from their web site.

6 | INTEGRATION PLATFORMS REVIEW

In this section, we analyse the run-time systems of the selected integration platforms to answer the third research question (RQ3). This analysis is guided by the evaluation framework previously introduced in Section 4 and aims to infer the values for the properties in the message processing and fairness execution dimensions. Results are summarised in Table 5 and discussed in the following sections. It is important to note that every selected integration platform was developed using Java programming language, which may result in the same values for different platforms.

6.1 | Message processing

In the following, we discuss how the run-time systems of the selected integration platforms meet the properties that can improve the efficiency in message processing. None of the platforms have their run-time system endowed with features to take advantage of multicore design. Mule uses the integration pattern⁴ Scatter Gather to execute tasks concurrently and independently. Camel implements integration patterns including Multicast, Splitter, and Aggregator, each one of these

TABLE 3 Integration platforms collected from the selected articles and reports

Integration Platform	Open Source	EIP	P&F	References
Action	X	X	X	29
Adaptris	X	X	X	65
Adeptia	X	X	X	29,70
AdroitLogic	✓	X	X	42
Apache Camel	✓	✓	✓	3,55,57,62,69
Apache Synapse	✓	X	✓	42
Attunity	X	X	X	29,65
Azuqua	X	X	X	65
Babelway	X	X	X	65
Built.io	X	X	X	29,65
Celigo	X	X	X	29,65
DBSync	X	X	X	29
Dell Boomi	✓	X	X	29,65,70
Elastico.io	X	X	X	65
Fiorano ESB	X	✓	X	42
Flowgear	X	X	X	65,70
Fujitsu	X	X	X	29
Guaraná	✓	✓	✓	3,57,59,63,64,69
Fuse	✓	✓	✓	42,54
IBM	X	X	X	29,41,42,54,65,66,70
Informatica	X	X	X	29,65
Jitterbit	✓	✓	✓	29,65,70
Microsoft	X	X	X	29,42,54,65,66
Moskitos	X	X	X	29
Mule	✓	✓	✓	3,29,42,54,57,65
Oracle	X	X	X	29,42,54,65,66
Petals	✓	✓	✓	54
SAP	✓	X	X	29,54,65
Scribe Software	X	X	X	29,65,70
ServiceMix	✓	✓	✓	54
Skyvva	X	X	X	65
SnapLogic	X	X	X	29,65,70
Software AG	X	X	X	65
Sonic ESB	X	✓	X	42
Spring Integration	✓	✓	✓	3,57
Talend	✓	X	X	42,65
TerraSky	X	X	X	29,65
Tibico	X	X	X	42,54,65
Vigence	X	X	X	65
Workato	X	X	X	70
WSO2	✓	✓	✓	42,54,61
Yourede	X	X	X	29,65

Abbreviation: EIP, enterprise integration patterns.

patterns providing a custom thread pool. Jitterbit has a feature that splits up the source data for parallel processing; each part is processed in isolation, and it is possible to process several parts in parallel. However, none of the platforms take advantage of multicore programming to execute integration solutions.

TABLE 4 Selection of the open-source integration platforms

Integration Platform	References
Apache Camel	3,55,57,62,69
Guaraná	3,57,59,63,64,69
Fuse	42,54
Jitterbit	29,65,70
Mule	3,29,42,54,57,65
Petals	54
ServiceMix	54
Spring Integration	3,57
WSO2	42,54,61

Regarding thread pool configuration, every run-time system is endowed with a feature to define a limit for the number of threads to be created during run time. Petals and Jitterbit do not provide information that allows evaluated them regarding this property, while Guaraná uses a thread pool with fixed number of threads.

Mule allows the configuration of threading profiles in three different ways: configuration, connector, and flow. Threading at configuration sets default threading profiles for all tasks. Threading at connector sets a threading profile for specific tasks, for example, one profile for tasks to receive messages and another to dispatch messages. Threading at flow sets a threading profile for a sequential flow of tasks.⁷² Both Camel and ServiceMix offer a fine-grained configuration where it is possible to tweak individual thread pools and have more coarse-grained configuration with fall back to global settings; furthermore, it should be possible to define a set of rules which matches the thread pool to a given source, ie, task or group of tasks.⁷³ Every run-time system have subclasses that extend classes of the Java language, whose methods allow to configure the thread pool. Camel and ServiceMix allow configuring settings of thread pool, whereas WSO2 has a file property that contains parameters for thread pool configuration.⁷⁴

Regarding the type of message storage in processes, Mule, Camel, Spring Integration, Fuse, ServiceMix, and Jitterbit can store data in-memory and in-disc. ServiceMix and WSO2 do not provide information that allows them to be evaluated regarding this property, and Guaraná stores messages only in-memory. Mule allows for storing data in-memory or in-disc for eventual retrieval, such as recovered data from processing into tasks like filters, routers, and other ones that need to store stateful messages. In the case of in-memory storage, Mule allows for storing data in a local memory of the run-time system, in which messages are dropped during shut-down of the run time. In the case of a persistent store, Mule persists data when explicitly configured to do that. In a standalone Mule run-time system, Mule creates a default persistent store in the file system.⁷² Mule allows for dealing with streamed data by configuring an initial memory buffer size of 512 kB; if the stream is larger than this, Mule creates a temporary file in-disc to store the contents without overflowing memory; if the stream is larger than 512 kB, the buffer is expanded to a default increment size of 512 kB until it reaches the configured maximum buffer size; when the stream exceeds this limit, the integration solution fails. Camel and ServiceMix allow for saving messages in a persistent store that can be a file or a database.⁷ Camel supports strategies to deal with streams; it can buffer all messages in an unbounded buffer or choose to keep only the latest or oldest message and drop all the others. Spring Integration defines a Message Store pattern, which allows components to store messages typically in some type of persistent store, in addition to the capability of buffering messages.⁷⁵ Fuse offers a number of different mechanisms for persistence besides the default message store. By default, a hybrid system that couples a data logs for message storage and a reference store for quick retrieval is used; another option allows for distributing the messages across multiple message stores or a file-based message store that maintain indexes into log files holding the messages that can be used. Additionally, Fuse supports the use of relational databases as a message store through the Java Database Connectivity feature, in which the persistence adaptor may be either coupled with a high data log or used in standalone mode.⁷⁶ To facilitate a rapid access to the content of the log, the message store constructs metadata to index the data embedded. Jitterbit is also able to persist message in-disc.⁷⁷

The execution of integration solutions in a distributed way makes the run-time system more suitable for cloud computing environments,⁷⁸ by providing greater processing power for tasks that can be processed in parallel across multiple virtual machines. However, it is important that the data transfer time from one machine to another minimises the total message processing time since the total processing time increases, and the distributed processing will be detrimental to

TABLE 5 Run-time systems comparison

Dimension	Property	Mule	Camel	Spring	Fuse	Integration Platforms	Petals	Jitterbit	WSO2	Guaraná
Message Processing	Designed for multicore	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	N/A	<i>no</i>	N/A	<i>no</i>
	Thread pool configuration	<i>limited</i>	<i>limited</i>	<i>limited</i>	<i>limited</i>	<i>limited</i>	N/A	N/A	<i>limited</i>	<i>fixed</i>
	Type of message storage in flow	<i>hybrid</i>	<i>hybrid</i>	<i>hybrid</i>	<i>hybrid</i>	<i>memory</i>	<i>memory</i>	<i>yes</i>	<i>memory</i>	<i>memory</i>
	Distributed process execution	<i>yes</i>	<i>yes</i>	<i>yes</i>	<i>yes</i>	<i>yes</i>	<i>yes</i>	<i>yes</i>	<i>yes</i>	<i>yes</i>
	Thread pool creation	<i>static</i>	<i>static</i>	<i>static</i>	<i>static</i>	<i>static</i>	N/A	<i>static</i>	<i>static</i>	<i>static</i>
Fairness Execution	Starvation detection	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	N/A	<i>no</i>	<i>no</i>
	Thread pool policy	<i>fifo</i>	<i>fifo</i>	<i>fifo</i>	<i>fifo</i>	<i>fifo</i>	<i>fifo</i>	<i>fifo</i>	<i>priority</i>	<i>fifo</i>
	Task computational complexity	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	<i>no</i>	N/A	N/A	<i>no</i>
	Execution Model	<i>process-based</i>	<i>process-based</i>	<i>process-based</i>	<i>process-based</i>	<i>process-based</i>	<i>process-based</i>	<i>process-based</i>	<i>process-based</i>	<i>task-based</i>
	Throttling controller	<i>yes</i>	<i>no</i>	<i>yes</i>	<i>no</i>	<i>no</i>	<i>no</i>	<i>yes</i>	<i>no</i>	<i>no</i>

N/A, *not available*

the performance of the run-time system. The ability to distribute the execution of tasks among several virtual machines is present in every run-time system analysed, except for Guaraná.

Mule has a virtual run-time system composed of multiple nodes, which ensure high system availability to perform distributed processing. It is possible to configure a cluster in Mule for an integration solution to maximise performance using a profile of performance. By implementing the performance profile for specific applications within a cluster, it is also possible to maximise the scalability of the deployments while deploying applications with different performance and reliability requirements in the same cluster.⁷² Camel, Fuse, and ServiceMix bring different technologies to allow their integration solutions to be scalable and to distribute the load among different instances, such as load balancing, clustering, and cloud computing. The load balancing approach allows for distributing the load among different *proxys*. Clustering can be achieved by means of one or several instances of the run-time system running on the same machine or distributed in virtual machines in a cloud computing environment. For cloud computing, it is possible to consume or push messages to Cassandra NOSQL database.^{73,79} Spring Integration provides a consistent model for intraprocess and interprocess messaging implemented using Java Message Service.⁸

Petals distributes its processes in a static and dynamic way. Statically, no new node can be added to a running Petals cluster. Dynamically, this distribution may be updated regularly, so new nodes can be added to a running Petals cluster.⁸⁰ Jitterbit provides high availability and load balancing of integration operations across run-time systems within a group. This platform is automatically scaled within the cloud as necessary and does not require adding new run-time systems to expand capacity.⁷⁷ WSO2 implements a distributed process by means of two models.

The first model consists of two subcluster domains as a worker domain and a management domain. These subdomains take up loads according to a defined load balancing algorithm and autoscales according to the load on its nodes. The second model consists of a single cluster, in which a selected node works as both a worker and a manager. This worker node requires two load balancers and has to be configured in read/write mode. The others are set up in read-only mode. The management node also should be a well-known member in the nonmanagement worker nodes so that state replication and cluster messaging works.⁷⁴

Regarding thread pool creation, none of the analysed run-time systems is able to dynamically create pools of threads to optimise task execution strategies from the analysis of the flow of messages in the integration solution. In Camel, it is possible to define a set of rules that matches which thread pool a given source should use. Petals does not provide information that allows it to be evaluated regarding this property.

From the analysis of the properties that may have an impact on message processing, we observe that Mule, Camel, and Jitterbit are advancing in terms of parallel programming, although none of them has actually benefited from multicore programming. There has been a growing need for a mechanism to run-time system that gives software engineer a simple yet effective way to make use of multiple processors in a clean scalable manner. It is need to enable the run-time systems to automatically scale to make use of the number of available processors. The multithreading programming must be optimised for situations in which the run-time system is able to execute portions of an integration solution simultaneously, with each part executing on its own CPU. This can be used to significantly speed up the execution of some types of workflows that presented a set of tasks that can be processed in parallel. Parallel programming should be thought of as a possible improvement in integration platforms because it offers a way to significantly improve integration solution performance. The majority of them already manages threads but is not still endowed with the elasticity feature for configuration of thread pools, in which the size of these computational resources changes proportionally to the demand of tasks at run time. Most of them are equipped with the ability to deal with large volumes of data, including features to store data either in-memory or in-disc. In addition, they advanced the issue of exploiting the benefits of distributed processing; except for Guaraná, every run-time system is endowed with the feature for distributing the execution of tasks among several virtual machines. None of them is equipped with the ability to create thread pools dynamically, demanding the expertise of software engineers to create thread pools at design time.

6.2 | Fairness execution

In the following, we discuss how the run-time systems of the selected integration platforms are meeting the properties that can improve a fair execution of tasks of an integration solution.

Regarding starvation detection, none of the platforms is endowed with the ability to detect tasks that are not executed within an accepted time frame, except for Jitterbit, which does not provide information that allows to evaluate it regarding to this property. The absence of support to this feature can lead to a risk of tasks to remain waiting forever for a thread to

be assigned to it. Every run-time system can monitor the execution of integration solutions, so that to detect bottlenecks that may appear during execution. Mule can use the management console to monitor the health of the run-time system, ie, see which flows are running or stopped, and determine memory usage, which can be a clue for bottleneck detection. In this platform, it is also possible to view detailed information about the integration platform, including alerts, memory usage, threads, pools, files, operating system resources, and run-time system settings.⁷² Camel has extensive support for Java Management Extensions to allow monitoring and controlling executions of integration solutions.⁷³ Spring Integration allows to monitor message sources, enable metrics, to detect bottlenecks. Such metrics can count and measure the number of failed sent message, the mean message sent rate, the number of messages in queue, and the number of active threads.⁷⁵ Fuse and WSO2 use Java Management Extensions to monitor and manage resources that may themselves turn into bottlenecks, such as memory allocation, thread utilisation, data input and output operations, CPU consumption, and request processing time.^{76,81} Jitterbit provides an interface that allows to monitor every integration process to catch errors.⁷⁷ Petals provides metrics through a control development kit, such as current, maximum and minimum number of active threads, response time of tasks, and the execution time of a task.⁸⁰

Regarding thread pool policy, the heuristics adopted in every run-time system is FIFO, except for WSO2 that adopts a priority-based policy. The priority of tasks is an alternative implementation that allows tasks to be ordered within the channel based upon a priority. To prioritise the execution of tasks, WSO2 uses a Java class, which executes sequences of tasks with a given priority. This approach allows for software engineers to control the resources allocated to execute sequences and prevent high priority tasks from getting delayed and dropped.⁷⁴ This class is backed by a custom implementation which has multiple internal queues for handling separate priorities. The scheduling policy impacts on the total processing time of a message since it is possible to give priority to tasks that need to be executed first or more frequently, avoiding them to wait in a queue for a time larger than necessary before being assigned to threads.

Regarding task computational complexity, none of the run-time systems is endowed with a feature to take into account the computational complexity of tasks in the assignment of threads, except for Jitterbit and WSO2, which do not provide information that allows to evaluate them regarding this property.^{52,82}

Regarding to execution model, except Guaraná, the majority of the run-time systems adopt a process-based execution model. In spite of most of platforms adopt this model, they can process message synchronously or asynchronously. The synchronous approach is used to process messages in the same thread that initially has received the message. After the integration solution receives a message, all processing, including the processing of the response, is done by the same thread. The asynchronous approach uses a queue to keep tasks that wait an available thread. In this case, a thread checks the task queue, then catch a task to execute. Thus, in the processing of a message, different threads can execute the tasks of the integration solution.^{13,72,73,76,83}

Regarding to throttling controller, Mule, Camel, Spring Integration, Jitterbit, and WSO2 allow to control the rate of incoming messages at input ports; this type of control is absent in Guaraná; Fuse, ServiceMix, and Petals do not provide information that allows to evaluate them regarding this property. Camel and Fuse support the control of inbound messages by implementing an integration pattern that allows the configuration of the size of the queue that connects the message producer and the polling consumer; or to block any message producer if the internal queue is full. In Mule, it is possible to arrange for tasks to actively consume messages at regular intervals of time.⁷² Spring Integration and Jitterbit use a Java class for the same purpose.^{75,77} In WSO2, a polling inbound pattern allows for polling a given message source at regular intervals of time. The polling inbound pattern periodically checks for new messages in the message source, and if there are messages available, they are loaded into the integration solution.¹³

From the analysis of the properties that may have an impact on the fairness execution of the run-time systems, we observe that the integration platforms are similar. None of them is endowed with features that allow to detect starvation or consider task computational complexity. Most of them follow the FIFO policy for scheduling of tasks; adopt the process-based execution model; and have the ability of inbound control of messages. In this dimension, WSO2 differs from others platforms because it is the single platform endowed with the ability to prioritise to task execution.

7 | FUTURE RESEARCH DIRECTIONS

In this section, we discuss quality attributes for improving the performance of integration platforms of applications. Performance has become a critical issue in the cloud computing environment⁵⁰; thus, enterprises must take it into account to improve their integration solutions. As a starting point, it is possible to look at other research fields and get inspirations from them to solve similar problems, so that these ideas can be studied and adapted to enterprise application integration

TABLE 6 Issues of research

Dimension	Property
Message processing	Designed for multicore
	Thread pool configuration
	Thread pool creation
Fairness execution	Starvation detection
	Thread pool policy
	Task computational complexity

in the context of cloud computing. Table 6 summarises the properties that represent issues of research; for each of them, we discuss incipient studies identified in the literature, which permeate similar issues in different areas of knowledge, reiterating the pertinence of the focus on performance and the interest of the scientific community in addressing them.

7.1 | Directions for message processing

Software engineers seek to develop algorithms that can take full advantage of multicore in order to achieve a high level of parallelism and an overall high performance. The way algorithms are written strongly impact the success of multicore technology.⁸⁴ Multicore design is a property which should be present in run-time systems bearing in mind that the technological advances already offer resources to extend parallel programming. As to hardware, we can cite the modern technology of graphics processing units (GPU), which demands tens of thousands of concurrent threads to fully utilise the massive amount of processing resources.⁸⁵ GPUs hold massive parallel computing capabilities with the potential of accelerating computationally intensive algorithms.⁸⁶ Measurements of thread pool throughput and measurements of thread utilisation in combination with analysis of prior thread pool resizing actions can be used to determine the increase or decrease of the number of threads from a thread pool in a current resizing action.⁴⁸ An exponential moving average scheme that adjusts the idle time-out period and thread pool size to adapt the system to the changing environment can be used to predict the number of threads and thread pool management, ensuring better response time and CPU usage.⁸⁷ Throughput degradation can be minimised by means of the creation of threads, based on the estimation of the range of threads needed, found via task arrival times and task processing times.⁸⁸ Measures of request frequencies can be used to dynamically optimise thread pool size, using nonblocking synchronisation primitives offering advantages of scalability and liveliness.⁸⁹ It is necessary to advance in methods of thread pool configuration and creation, which allow dynamic thread pool suitable sizing for use in multithreaded processing environment such as a distributed data grid, providing a rapid and responsive adjustment of thread pool size in response to changes in workload and processor availability.

The previous studies motivate us to go deeper into the research that explores the advantages of multicore in the context of integration platforms, as well as the elastic configuration of the size of the thread pools and the dynamic creation of these pools by the run-time system during run time. These improvements would increase message processing and, consequently, allow for better performance on the integration platforms in the context of cloud computing.

7.2 | Directions for fairness execution

The implementation of multiple thread pools based on a distribution of service times can avoid starvation and achieve concurrent processing, decreasing the response time, and reducing the waiting time in the execution of tasks.⁹⁰ Studies indicate that task scheduling and resource allocation can be optimised by means of algorithms such as differential evolution algorithms based on the proposed cost and time models in cloud computing.⁹¹ Algorithms combining different policies, like shortest-job-first and round-robin schedulers,⁹² or other meta-heuristics techniques based on swarm intelligence and bio-inspired techniques could help to choose a suitable approach for better schemes for scheduling according to the application.⁹³

It is essential to balance the execution time of all tasks in a workflow running in parallel in order to achieve the best possible use of computing resources. One of the possible solutions to deal with the heterogeneity of the underlying platform is to use a model that to consider different execution times for loop iterations of the program. Thus, these execution times are taken into account to select the processing units according to its performance characteristics, as well as, to determine the number of processing units that are used simultaneously.⁹⁴ An efficient and economical way of using computational resources is to adopt policies and approaches for deciding the task granularity at run time based on computational resource utilisation constraints, quality of service requirements, and the average task deployment metrics.⁵² The prediction of the

quantity of computational resources that are needed for a reconfigurable architecture to suit task granularity can be used to make compatible resources and tasks.⁸² Computational complexity should also be considered in the scheduling process to achieve better allocation of computational resources.

These incipient research reflects the concern of the scientific community regarding performance. Such ideas still have to be developed, applied, and experimented on the run-time systems of integration platforms in the context of cloud computing. Thus, these and other studies point to solutions that can endow platforms with features that provide an efficient message processing and fairness execution of tasks, and, consequently, the run-time systems will achieve better performance in the execution of integration solutions in cloud computing.

8 | THREATS TO VALIDITY

We evaluated the factors that influence the obtained results and the main limitations of the survey, which are describe below. The screening research procedure is based on the subjective evaluation of a single team, so it may not be representative to other groups of software engineers. Besides, this procedure is based only on the documentation of the platforms; thus, it is possible that practical experience in use of these platforms would allow to identify new quality attributes that have an impact on the performance of run-time systems as well. We have chosen screening because this is the faster and least costly form of feature analysis.³⁶ Scopus was the single database searched; thus, it is possible that our survey had not found other existing platforms. We have chosen Scopus because it is one of the leading databases of scholarly impact, offering significantly journal coverage and the authors found in Scopus have higher h-indexes.⁹⁵

9 | CONCLUSIONS

Cloud computing has provided several services to companies, which offer the chance to leverage their business processes. These services frequently require the integration of different applications that compose the software ecosystem. Thus, companies need to rely on integration platforms that enable better performance. Integration platforms are specialised software tools that allow for keeping information on all these systems consistent and synchronised. Usually, an integration platform is composed of a domain-specific language, a development toolkit, a run-time system, and monitoring tools. The run-time system is responsible for running integration solutions and, therefore, its performance is what most frequently drives the decision of companies when choosing an integration platform.

In this article, we presented a survey on the run-time systems of integration platforms. We analysed properties that can have an impact on their performance when executing integration solutions. These properties are organised into two dimensions, namely, message processing and fairness execution. The former dimension deals with the efficiency of the run-time system to process a message, which refers to the improvement of the rate of messages processed in an integration solution. The latter dimension focuses on the assignment of threads to tasks to provide a minimum process time for a message in an integration solution.

We evaluated the run-time systems of nine different state-of-the-art open-source message-based integration platforms and identified issues to be resolved to improve the performance of their run-time systems from the perspective of message processing and fairness execution. It was possible to identify that run-time systems have advanced regarding efficiency of message processing since most of them allow the number of threads in a thread pool to be increased automatically during run time until a threshold defined at design time is reached; most of them store messages in-memory and in-disc; all of them are able to distribute the processing. On the other hand, none of them has been designed to take advantage of multicore and none of them is able to dynamically create thread pools. The platforms, however, have few features that allow the fair execution of tasks. None is able to detect tasks that have been waiting for a long time to be executed, ie, starvation detection; most of them use basic heuristics as scheduling policy to tasks; none of them recognises the computational complexity of these tasks; most of them adopt the process-based execution model; and only a small part of them has some kind of throttling controller to limit the incoming message rate.

Among the analysed run-time systems, Mule is the one that meets the most properties that positively impact performance. It is able to manage the size of the thread pool according to the demand of the tasks; to store the messages both in-memory and in-disc; to distribute the processing; to detect hotspots at run time; and to control the rate of incoming messages.

Based on these findings, we suggest directions to achieve better performance results with the run-time systems of the integration platforms. Considering the message processing dimension the run-time system can be improved regarding its design for multicore, thread pool configuration, and thread pool creation. As for fairness execution, they can be improved regarding starvation detection, thread pool policy, and task computational complexity.

ACKNOWLEDGEMENTS

This work was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) under grants 73318345415 and 88881.119518/2016-01 and the Research Support Foundation of the State of Rio Grande do Sul (FAPERGS) under grant 17/2551-0001206-2. We would like to thank Dr Rafael Corchuelo and Dr Inma Hernández from the University of Seville (Spain) and Ms Elizabeth Thornton Rush from the Pennsylvania State University (United States) for their helpful comments in earlier versions of this article.

ORCID

Rafael Z. Frantz  <https://orcid.org/0000-0003-3740-7560>

REFERENCES

- Messerschmitt DG, Szyperski C. *Software Ecosystem: Understanding an Indispensable Technology and Industry*. Cambridge, MA: MIT Press; 2003.
- Manikas K. Revisiting software ecosystems research: a longitudinal literature study. *J Syst Softw*. 2016;117:84-103.
- Frantz RZ, Corchuelo R, Roos-Frantz F. On the design of a maintainable software development kit to implement integration solutions. *J Syst Softw*. 2016;111:89-104.
- Hohpe G, Woolf B. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA: Pearson Education; 2004.
- Alexander C, Ishikawa S, Silverstein M, Jacobson M, Fiksdahl-King I, Angel S. *A Pattern Language: Towns, Buildings, Construction*. New York, NY: Oxford University Press; 1977.
- Dossot D, D'Emic J, Romero V. *Mule in Action*. Greenwich, CT: Manning Publications Co; 2014.
- Ibsen C, Anstey J. *Camel in Action*. Greenwich, CT: Manning Publications Co; 2010.
- Fisher M, Partner J, Bogoevici M, Fuld I. *Spring Integration in Action*. Greenwich, CT: Manning Publications Co; 2012.
- Russell J, Cohn R. *Fuse ESB*. Stoughton, WI: Book on Demand; 2012.
- Konsek H. *Instant Apache ServiceMix How-to*. Birmingham, UK: Packt Publishing; 2013.
- Russell J, Cohn R. *Petals ESB*. Stoughton, WI: Book on Demand; 2012.
- Russell J, Cohn R. *Jitterbit Integration Server*. Stoughton, WI: Book on Demand; 2012.
- Indrasiri K. Introduction to WSO2 ESB. In: *Beginning WSO2 ESB*. Berkeley, CA: Apress; 2016:1-16.
- Frantz RZ, Corchuelo R, Molina-Jiménez C. A proposal to detect errors in enterprise application integration solutions. *J Syst Softw*. 2012;85(3):480-497.
- St. Laurent AM. *Understanding Open Source & Free Software Licensing: Guide to Navigating Licensing Issues in Existing & New Software*. Sebastopol, CA: O'Reilly Media Inc; 2004.
- Mell P, Grance T. *The NIST Definition of Cloud Computing*. Gaithersburg, MD: Information Technology Laboratory; 2011.
- Khoubati K, Themistocleous M, Irani Z. Evaluating the adoption of enterprise application integration in health-care organizations. *J Manag Inf Syst*. 2006;22(4):69-108.
- Ebert N, Weber K, Koruna S. Integration platform as a service. *Bus Inf Syst Eng*. 2017;59(5):375-379.
- Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Futur Gener Comput Syst*. 2009;25(6):599-616.
- da Silva Dias A, Nakamura LHV, Estrella JC, Santana RHC, Santana MJ. Providing IaaS resources automatically through prediction and monitoring approaches. Paper presented at: 2014 IEEE Symposium on Computers and Communication (ISCC); 2014; Funchal, Portugal.
- Ma L, Agrawal K, Chamberlain RD. A memory access model for highly-threaded many-core architectures. *Futur Gener Comput Syst*. 2014;30:202-215.
- Li Q, Wang Z, Li W, Cao Z, Du R, Luo H. Model-based services convergence and multi-clouds integration. *Comput Ind*. 2013;64(7):813-832.
- Caseau Y. Self-adaptive middleware: supporting business process priorities and service level agreements. *Adv Eng Inform*. 2005;19(3):199-211.
- Schulte S, Janiesch C, Venugopal S, Weber I, Hoenisch P. Elastic business process management: state of the art and open challenges for BPM in the cloud. *Futur Gener Comput Syst*. 2015;46:36-50.
- Rosati P, Fox G, Kenny D, Lynn T. Quantifying the financial value of cloud investments: a systematic literature review. Paper presented at: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom); 2017; Hong Kong, China.

26. Brahmi Z, Gharbi C. Temporal reconfiguration-based orchestration engine in the cloud computing. Paper presented at: International Conference on Business Information Systems (ICBIS); 2014; Larnaca, Cyprus.
27. Pezzini M, Lheureux BJ. Integration platform as a service: moving integration to the cloud. Gartner Inc; 2011. G00210747.
28. Sharma S. Ovum decision matrix: selecting an integration PaaS (iPaaS), 2015-2016. Ovum Consulting; 2015.
29. Guttridge K, Pezzini M, Golluscio E, Thoo E, Iijima K, Wilcox M. Magic quadrant for enterprise integration platform as a service. Gartner Inc; 2017. G00304070.
30. Pezzini M, Natis YV, Malinverno P, et al. Magic quadrant for enterprise integration platform as a service, worldwide. Gartner Stamford; 2015. G00270939.
31. Sharma S. Ovum decision matrix highlights the growing importance of iPaaS and API platforms in hybrid integration. Ovum Consulting; 2017. it0022-000670.
32. Harman M, Lakhota K, Singer J, White DR, Yoo S. Cloud engineering is search based software engineering too. *J Syst Softw.* 2013;86(9):2225-2241.
33. Boillat T, Legner C. Why do companies migrate towards cloud enterprise systems? A post-implementation perspective. Paper presented at: 2014 IEEE 16th Conference on Business Informatics; 2014; Geneva, Switzerland.
34. Kitchenham B, Linkman S, Law D. DESMET: a methodology for evaluating software engineering methods and tools. *Comput Control Eng J.* 1997;8(3):120-126.
35. Garousi V, Felderer M, Mäntylä MV. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Inf Softw Technol.* 2018;1-22.
36. Kitchenham BA. Evaluating software engineering methods and tools, part 7: planning feature analysis evaluation. *ACM SIGSOFT Softw Eng Notes.* 1997;22(4):21-24.
37. Ogawa RT, Malen B. Towards rigor in reviews of multivocal literatures: applying the exploratory case study method. *Rev Educ Res.* 1991;61(3):265-286.
38. Strauss A, Corbin JM. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques.* Thousand Oaks, CA: Sage Publications Inc; 1990.
39. Corchuelo R, Frantz RZ, González J. Una comparación de ESBs desde la perspectiva de la integración de aplicaciones. Paper presented at: Jornadas de Ingeniería del Software y Bases de Datos (JISBD); 2008; Gijón, Spain.
40. García-Jiménez FJ, Martínez-Carreras MA, Gómez-Skarmeta AF. Evaluating open source enterprise service bus. Paper presented at: 2010 IEEE 7th International Conference on E-Business Engineering (ICEBE); 2010; Shanghai, China.
41. Palanimalai S, Paramasivam I. An enterprise oriented view on the cloud integration approaches—hybrid cloud and big data. *Procedia Comput Sci.* 2015;50:163-168.
42. Bhadoria RS, Chaudhari NS, Tomar GS. The performance metric for enterprise service bus (ESB) in SOA system: theoretical underpinnings and empirical illustrations for information processing. *Inf Syst.* 2017;65:158-171.
43. Ritter D, May N, Rinderle-Ma S. Patterns for emerging application integration scenarios: a survey. *Inf Syst.* 2017;67:36-57.
44. Janetschek M, Prodan R, Benedict S. A workflow runtime environment for manycore parallel architectures. *Futur Gener Comput Syst.* 2017;75:330-347.
45. Gleyzer G, Howes J, inventors; Oracle International Corp, assignee. System and method for supporting dynamic thread pool sizing in a distributed data grid. US patent 9,547,521 B2. January 17, 2017.
46. Balko S, Barros A. In-memory business process management. Paper presented at: 2015 IEEE 19th International Enterprise Distributed Object Computing Conference (EDOC); 2015; Adelaide, Australia.
47. He W, Xu LD. Integration of distributed enterprise applications: a survey. *IEEE Trans Ind Inform.* 2014;10(1):35-42.
48. Nazeer S, Bahadur F, Khan MA, Hakeem A, Gul M, Umar AI. Prediction and frequency based dynamic thread pool system a hybrid model. *Int J Comput Sci Inf Secur.* 2016;14(5):299-30.
49. Burns DW, Allen JD, Upton MD, Boggs DD, Kyker AB, inventors; Intel Corp, assignee. Determination of approaching instruction starvation of threads based on a plurality of conditions. US patent 6,651,158. November 18, 2003.
50. Anwar N, Deng H. Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments. *Future Internet.* 2018;10(1):1-23.
51. Chen W, Deelman E. Workflow overhead analysis and optimizations. In: Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science (WORKS); 2011; Seattle, WA.
52. Muthuvelu N, Vecchiola C, Chai I, Chikkannan E, Buyya R. Task granularity policies for deploying bag-of-task applications on global grids. *Futur Gener Comput Syst.* 2013;29(1):170-181.
53. Frantz RZ, Corchuelo R, Arjona JL. An efficient orchestration engine for the cloud. Paper presented at: 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom); 2011; Athens, Greece.
54. Myroshnychenko K. *Maturidade de Plataformas SOA Open Source vs Proprietários* [master's thesis]. Lisbon, Portugal: Departamento de Engenharia Informática e Sistemas de Informação, Universidade Lusófona de Humanidades e Tecnologias; 2013.
55. Emmersberger C, Springer F. Tutorial: open source enterprise application integration - introducing the event processing capabilities of apache camel. In: Proceedings of the 7th ACM International Conference on Distributed Event-based Systems (DEBS); 2013; Arlington, TX.
56. Le Hors AJ, Speicher S. The linked data platform (LDP). In: Proceedings of the 22nd International Conference on World Wide Web (WWW); 2013; Rio de Janeiro, Brazil.

57. Klein MJ, Sawicki S, Roos-Frantz F, Frantz RZ. On the formalisation of an application integration language using Z notation. In: Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS); 2014; Lisbon, Portugal.
58. Penciu D, Durupt A, Belkadi F, Eynard B, Rowson H. Towards a PLM interoperability for a collaborative design support system. *Procedia CIRP*. 2014;25:369-376. Conference on Industrial Product-Service Systems.
59. Sequeira FR, Frantz RZ, Yevseyeva I, Emmerich MTM, Basto-Fernandes V. An EAI based integration solution for science and research outcomes information management. *Procedia Comput Sci*. 2015;64:894-901.
60. Hernández I, Sawicki S, Roos-Frantz F, Frantz RZ. Cloud configuration modelling: a literature review from an application integration deployment perspective. *Procedia Comput Sci*. 2015;64:977-983.
61. Kurniawan K, Ashari A. Service orchestration using enterprise service bus for real-time government executive dashboard system. Paper presented at: 2015 International Conference on Data and Software Engineering (ICoDSE); 2015; Yogyakarta, Indonesia.
62. Ritter D, May N, Sachs K, Rinderle-Ma S. Benchmarking integration pattern implementations. Paper presented at: International Conference on Distributed and Event-based Systems (DEBS); 2016; Irvine, CA.
63. Belusso CLM, Sawicki S, Roos-Frantz F, Frantz RZ. A study of Petri Nets, Markov chains and queueing theory as mathematical modelling languages aiming at the simulation of enterprise application integration solutions: a first step. *Procedia Comput Sci*. 2016;100:229-236.
64. Kraissig AR, Welter FC, Haugg IG, et al. Mathematical model for simulating an application integration solution in the academic context of Unijui University. *Procedia Comput Sci*. 2016;100:407-413.
65. Ebert N, Weber K. Integration platform as a service in der praxis: eine bestandsaufnahme. In: *Multikonferenz Wirtschaftsinformatik (MKWI)*. Ilmenau, Germany: Technische Universität Ilmenau; 2016:1675-1985.
66. Bauer KS, Kuznetsov DY, Pominov AD. Designing the search service for enterprise portal based on Oracle universal content management. *J Phys Conf Ser*. 2017;803(1):1-5.
67. Tirkey A, Gary KA. Curricular change management with Git and Drupal: a tool to support flexible curricular development workflows. Paper presented at: 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA); 2017; London, UK.
68. Sellaro DF, Frantz RZ, Hernández I, Roos-Frantz F, Sawicki S. Task scheduling optimization on enterprise application integration platforms based on the meta-heuristic particle swarm optimization. In: Proceedings of the 31st Brazilian Symposium on Software Engineering (SBES); 2017; Fortaleza, Brazil.
69. Ritter D, Forsberg FN, Rinderle-Ma S, May N. Optimization strategies for integration pattern compositions. In: Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems (DEBS); 2018; Hamilton, New Zealand.
70. Peyret H, Cullen A, Kramer A, Lynch D. The Forrester wave™: iPaaS for dynamic integration, Q3 2016. Forrester Inc; 2016.
71. Guttridge K, Pezzini M, Malinverno P, et al. Magic quadrant for enterprise integration platform as a service, worldwide. Gartner Inc; 2016. G00277000.
72. MuleSoft. *Mule 3.8.x Performance Tuning Guide*. White paper. 2017. Mule user guide. Accessed February 27, 2017.
73. Apache Camel. Design notes for threadpool configuration. Version 2.20. 2017. Accessed February 27, 2017.
74. WSO2 Enterprise Integrator Documentation. Configuring properties. Version 6.3.0. 2017. Accessed February 27, 2017.
75. Fisher M, Bogoevici M, Fuld I, et al. Spring integration reference manual. Version 5.1.0. 2017. Accessed February 27, 2017.
76. RedHat. Product documentation for Red Hat fuse 7.0: configuring broker persistence. 2017. Accessed February 27, 2017.
77. Jitterbit Success Central. Harmony 9.0: chunking. 2017. Accessed February 27, 2017.
78. Hwang K, Dongarra J, Fox GC. *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Waltham, MA: Morgan Kaufmann; 2013.
79. Anand S, Singh P, Sagar BM. Working with Cassandra database. In: *Information and Decision Sciences: Proceedings of the 6th International Conference on FICTA*. Singapore: Springer Nature Singapore Pte Ltd; 2018:531-538.
80. Petals Documentation. Configuring petals ESB: topology configuration. Version 5.1.0. 2017. Container user guide. Accessed February 27, 2017.
81. Synapse. Apache synapse - documentation. Version 2.0.0. 2017. Accessed February 27, 2017.
82. Sudarsanam A, Srinivasan M, Panchanathan S. Resource estimation and task scheduling for multithreaded reconfigurable architectures. In: Proceedings of the Tenth International Conference on Parallel and Distributed Systems (ICPADS); 2004; Newport Beach, CA.
83. Pandey C. *Spring Integration Essentials*. Birmingham, UK: Packt Publishing Ltd; 2015.
84. Sethi A, Kushwah H. Multicore processor technology - advantages and challenges. *Int J Res Eng Technol*. 2015;4(9):87-89.
85. Yoon MK, Kim K, Lee S, Ro WW, Annaram M. Virtual thread: maximizing thread-level parallelism beyond GPU scheduling limit. Paper presented at: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA); 2016; Seoul, South Korea.
86. Tang W, Feng W, Deng J, Jia M, Zuo H. Parallel computing for geocomputational modeling. In: *GeoComputational Analysis and Modeling of Regional Systems*. Cham, Switzerland: Springer International Publishing AG; 2018:37-54. *Advances in Geographic Information Science*.
87. Lee K-L, Pham HN, Kim H-s, Youn HY, Song O. A novel predictive and self - adaptive dynamic thread pool management. Paper presented at: 2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications (ISPA); 2011; Busan, South Korea.
88. Oh S-K, Kim J-S. A history-based dynamic thread pool method for reducing thread creation and removal overheads. *J Adv Navig Technol*. 2013;17(2):189-195.
89. Bahadur F, Naeem M, Javed M, Wahab A. FBOS: frequency based optimization strategy for thread pool system. *Nucleus*. 2014;51(1):93-107.
90. Shah R, Bahdur F, Hu N-U-A, Umer A, Shad MJ. *Implementation of Multiple Thread Pools Based on Distribution of Service Times*. Working paper. 2017.

91. Tsai J-T, Fang J-C, Chou J-H. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Comput Oper Res.* 2013;40(12):3045-3055.
92. Elmougy S, Sarhan S, Joundy M. A novel hybrid of shortest job first and round Robin with dynamic variable quantum time task scheduling technique. *J Cloud Comput.* 2017;6(1):1-12.
93. Singh P, Dutta M, Aggarwal N. A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowl Inf Syst.* 2017;52(1):1-51.
94. Cordes D, Heinig A, Marwedel P, Mallik A. Automatic extraction of pipeline parallelism for embedded software using linear programming. Paper presented at: 2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS); 2011; Tainan, Taiwan.
95. Powell KR, Peterson SR. Coverage and quality: a comparison of Web of Science and Scopus databases for reporting faculty nursing publication metrics. *Nurs Outlook.* 2017;65(5):572-578.

How to cite this article: Freire DL, Frantz RZ, Roos-Frantz F, Sawicki S. Survey on the run-time systems of enterprise application integration platforms focusing on performance. *Softw: Pract Exper.* 2018;1–20. <https://doi.org/10.1002/spe.2670>