


A Survey on Global Management View: Toward Combining System Monitoring, Resource Management, and Load Prediction

Rodrigo da Rosa Righi  · Matheus Lehmann · Marcio Miguel Gomes ·
Jeferson Campos Nobre · Cristiano André da Costa · Sandro José Rigo ·
Marcio Lena · Rodrigo Fraga Mohr · Luiz Ricardo Bertoldi de Oliveira

Received: 24 May 2018 / Accepted: 17 December 2018
© Springer Nature B.V. 2019

Abstract Today, enterprise applications impose more and more resource requirements to support an ascending number of clients and to deliver them an acceptable Quality of Service (QoS). To ensure such requirements are met, it is essential to apply appropriate resource and application monitoring techniques. Such techniques collect data to enable predictions and actions which can offer better system performance.

The final version of the present manuscript was approved by Marcio Lena on behalf of Dell Technologies.

R. da Rosa Righi (✉) · M. Lehmann · M. M. Gomes ·
J. C. Nobre · C. A. da Costa · S. Rigo ·
L. R. B. de Oliveira
Software Innovation Laboratory (SoftwareLab) - Applied
Computing Graduate Program, University of Vale do Rio
dos Sinos (UNISINOS), Av. Unisinos, 950 São Leopoldo,
RS, Brazil
e-mail: rrrighi@unisinos.br

M. Lehmann
e-mail: mblehmann@inf.ufrgs.br

M. M. Gomes
e-mail: marciomg@unisinos.br

J. C. Nobre
e-mail: jcnobre@unisinos.br

C. A. da Costa
e-mail: cac@unisinos.br

S. Rigo
e-mail: rigo@unisinos.br

L. R. B. de Oliveira
e-mail: luizbertoldi@unisinos.br

Typically, system administrators need to consider different data sources, so making the relationship among them by themselves. To address these gaps and considering the context of general networked-based systems, we propose a survey that combines a discussion about system monitoring, data prediction, and resource management procedures in a unified view. The article discusses resource and application monitoring, resource management, and data forecast at both performance and architectural perspectives of enterprise systems. Our idea is to describe consolidated subjects such as monitoring metrics and resource scheduling, together with novel trends, including cloud elasticity and artificial intelligence-based load prediction algorithms. This survey links the aforesaid three pillars, emphasizing relationships among them and also pointing out opportunities and research challenges in the area.

Keywords Performance · Monitoring metrics ·
Computing infrastructure · Resource monitoring ·
Resource management · Load prediction · Time series

M. Lena · R. F. Mohr
DELL, Av. Industrial Belgraf, 400, Eldorado do Sul, RS,
Brazil
e-mail: marcio.lena@dell.com

R. F. Mohr
e-mail: rodrigo.mohr@dell.com

1 Introduction

Performance is always an issue for enterprise applications, either to support a particular Return on Investment (ROI) or to provide Quality of Service (QoS) to end users [17, 68]. Enterprise applications are defined by Gartner¹ as a set of integrated computer systems designed to control the enterprise's core business. Today, are increasingly common applications that incorporate many functionalities, needing high-speed CPU and IO operations to provide ROI and QoS at acceptable levels. From the application perspective, complexity can be understood as another key challenge, where multiple software layers interconnect, possibly presenting an arbitrary dependency relationship and software engineering problems (such as lack of maintainability and system isolation). In addition, the hardware aspect is as important as the software one. Taking into account the behavior of the application (e.g., CPU- or IO-bound, including at least memory, disk, and network), a particular set of resource infrastructure must be provided to support this behavior adequately, so not being the bottleneck of the system execution [34, 40, 47].

Looking at both hardware and software perspectives, the correct use of resource management techniques is crucial to obtain performance. In this way, process scheduling and rescheduling [16, 34, 40], load balancing [49, 72], resource elasticity [6, 24, 25], and high-performance computing [47] appear as alternatives to handle performance at application launching. In general, the common approach idea is to work with one or more processes representing the application and a collection of resources (commonly referred as computational nodes) in such a way that the aforementioned techniques are used to reduce the time needed by the application to perform its tasks. To accomplish this, these four techniques need both software and hardware modeling. On the software side, an application modeled and implemented as building blocks - such as processes, threads, procedures, data buffers, and entry queues - can get high performance running in a multi core and multi thread parallel architecture compared to a software developed to run in a linear single thread structure. In the hardware panorama, new infrastructure resources must be delivered in an acceptable time

interval, mainly without blocking the current application execution. For example, it is not valid to enlarge the number of computational nodes with resource elasticity, but not preparing the application to benefit from this. Furthermore, when dealing with process scheduling, it is desirable to know information about the process behavior and the hardware features, so proposing a more process-resource accurate mapping.

In general, the conventional way to implement resource management techniques that bring significant performance improvements for the application considers the use of a useful back-end support of system monitoring [2, 21]. This system monitoring can be split into resource and application monitoring [3]. The first targets CPU load, memory usage, network inbound and outbound, disk usage or any other relevant resource-related metric by employing continuous or discrete data collection procedures [2]. The application monitoring, in its turn, analyzes the application behavior regarding computation and communication parts, then detaching execution patterns that help on deciding the current resource management technique to apply and under which conditions to do that [21]. In parallel to monitoring and resource management, we also envisage the wide use of artificial intelligence (AI) to support better performance and an easier system administration. Here, we emphasize the role of load prediction algorithms that work with the past to propose assumptions about the future of the application execution [14]. Most of the enterprise systems are characterized by a reactive behavior, where an action only takes place after an abnormal situation was perceived. In this context, load prediction works in favor of anticipating eventual problems. Such anticipation can enable actions to nullify or mitigate the occurrence of problems, both in an automated (e.g., using Artificial Intelligence) or manual (e.g., triggering a notification for the human administrator) way.

System monitoring, resource management, and load prediction present a strong relationship in favor of performance on enterprise systems. For example, it is possible to monitor the CPU load every minute, then using this historical data to forecast the CPU load in the next hour. This prediction can serve as input for the resource elasticity procedure, launching a Virtual Machine (VM) or a container to deliver resources before a real problem occurs. Our analysis of the literature did not find an article that presents

¹<https://www.gartner.com/it-glossary/enterprise-applications>

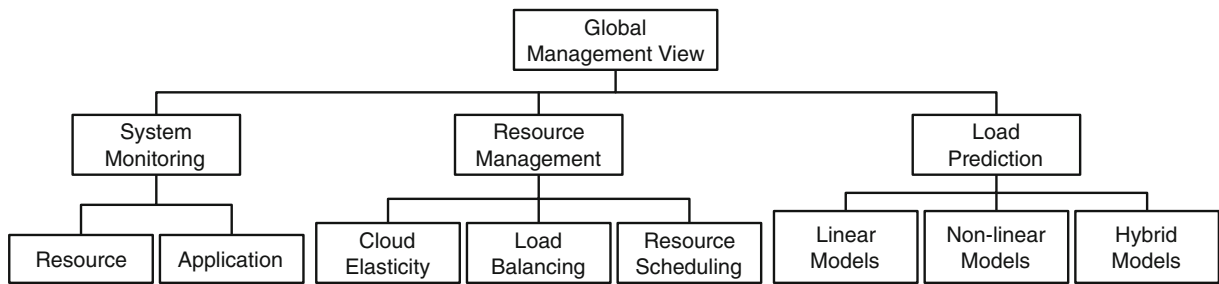


Fig. 1 Overview of the survey

the three thematics together in a single survey document. Thus, system administrators should consider different strategies, so making the relationship among them by themselves. Taking this gap as motivation and considering the scope of networked-based systems, this article presents a survey that discusses resource and application monitoring, resource management, and data forecast at both performance and architecture perspectives of enterprise systems [78].

Figure 1 depicts an overview classification about the topics discussed in the current survey. Our idea is to describe consolidated subjects such as monitoring metrics and resource scheduling, together with novel trends, like cloud elasticity and AI-based load prediction algorithms.

Considering the methodology on selecting the articles, we used three sets of queries, each one for a particular topic addressed in the survey: (i) system monitoring: “monitoring” AND (“system” OR “resource” OR “application” OR “cloud” OR “grid”) AND “computing”; (ii) resource management: “cloud computing” AND (“elasticity” OR “load balancing” OR “scheduling”) AND “high-performance” (iii); load prediction: “time series” AND (“prediction” OR “forecast”) AND (“ARIMA” OR “ANN” OR “hybrid”). Over the selected articles, we applied the following exclusion criteria: (i) articles greater than 4 pages; (ii) written in english; (iii) published in the last 15 years; (iv) duplicates removal. We did not apply any exclusion criteria concerning the nature of the study, including either theoretical approach and practical experiments. The consulted bases were ACM Digital Library,² Elsevier Science Direct,³ IEEE Xplore Digital Library⁴ and Springer Link.⁵

²<https://dl.acm.org>

³<https://www.sciencedirect.com>

⁴<http://ieeexplore.ieee.org>

⁵<https://link.springer.com>

As the main contribution, we prepared two sections to discuss the interconnections of the three pillars of the survey and potential research opportunities. The remainder of this article will first introduce the related work in Section 2. Sections 3, 4, and 5 present a description of system monitoring, resource management, and load prediction, respectively. In each one of these three sections, our idea is to first present theoretical content about section thematic, then highlighting significant initiatives about it afterward. Section 6 discusses the relationship between the three topics: system monitoring, resource management, and load prediction. We present in Section 7 presents the research and development opportunities regarding the three pillars focused on this article. Finally, Section 8 emphasizes the scientific contribution of the work and notes several challenges that we can address in the future.

2 Related Work

This section presents some initiatives that attempt to provide an overview of system monitoring, resource management or load prediction. Table 1 summarizes the surveys resulted from these initiatives, categorizing them according to the topics they cover. Next, we comment them briefly.

Several literature surveys cover resource and application monitoring as follows. Aceto et al. [2] analyze the state-of-the-art in cloud monitoring. The authors propose a two-axis taxonomy for the topic: one axis concerns the several motivations for monitoring cloud computing, the other axis comprises three dimensions: layers, abstractions level, tests, and metrics. Fatema et al. [21] introduce a taxonomy and analysis of a broad variety of monitoring tools, from general-purpose infrastructure ones to high-level application monitoring services. They identify practical capabilities that

Table 1 Summary of the related work

Survey	System	Resource	Load
	Monitoring	Management	Prediction
Aceto et al. [2]	✓		
Fatema et al. [21]	✓		
Alhamazani et al. [8]	✓		
Krauter et al. [34]	✓	✓	
Netto et al. [47]	✓	✓	
Galante and d. Bona [24]		✓	
Galante et al. [25]		✓	
Waraich [72]		✓	
Patel et al. [49]		✓	
Casavant and Kuhl [16]		✓	
Madni et al. [40]		✓	
Al-Dhuraibi et al. [6]		✓	✓
Seneviratne and Witharana [65]			✓
Sahi and Dhaka [62]			✓
Manvi and Shyam [42]			✓
Weingaärtner et al. [74]			✓
Amiri and Mohammad-Khanli [9]			✓

an ideal monitoring tool should possess. Alhamazani et al. [8] survey the commercial cloud monitoring tools, discussing research dimensions, design issues, and the state-of-the-art. The authors investigate the fundamental understanding of cloud resource as well as application provisioning and monitoring concepts. As a conclusion, they present future research directions for novel cloud monitoring techniques.

There are several efforts to organize the literature regarding resource management techniques. Because these techniques rely both on the current system state and predicting the future state, they can also cover, even if briefly, some aspects of system monitoring and load balancing. The surveys in resource management commonly address only one specific technique individually: cloud elasticity [6, 24, 25], load balancing [49, 72] or resource scheduling [16, 34, 40]. Based on the state-of-the-art mechanisms for each technique, they categorize the main properties of the solutions proposed together with their alternatives and discuss the open challenges not covered yet in the literature. More recently, Netto et al. [47] propose a survey of High-Processing Computing which covers multiple management techniques applied to a specific type of application.

As for load prediction, we present the following selection of surveys. Seneviratne and Witharana [65]

suggest a procedure for pre-processing the records in automatically generated log files. After, they discuss five prediction methods, including statistic based threshold, time series analysis, rule-based classification, Bayesian network models, and semi-Markov process models. According to Sahi and Dhaka [62], inaccuracy in workload prediction can lead to over-provisioning or under-provisioning of resources. The workload prediction differs from one environment to another because relevant parameters (e.g., bandwidth, storage, computing, and availability) vary according to the type of application. Other three studies [42, 74] and [9] describe, compare, and discuss profiling and forecasting models as well as techniques to estimate the amount of resource needed for a workload.

System monitoring, resource management, and load prediction have a high correlation regarding the performance of enterprise systems. However, as evidenced in Table 1, these topics are usually treated separately. In few occasions, two of the topics can be found in a single survey but explored without depth. The present article attempts to fill this gap by presenting a survey that discusses in detail resource and application monitoring, resource management and data forecast at both performance and architecture perspectives of enterprise systems.

3 System Monitoring

The efficient management of resources currently attracts significant interest. Monitoring data is an essential component to provide adequate Quality of Service (QoS) and Quality of Experience (QoE)

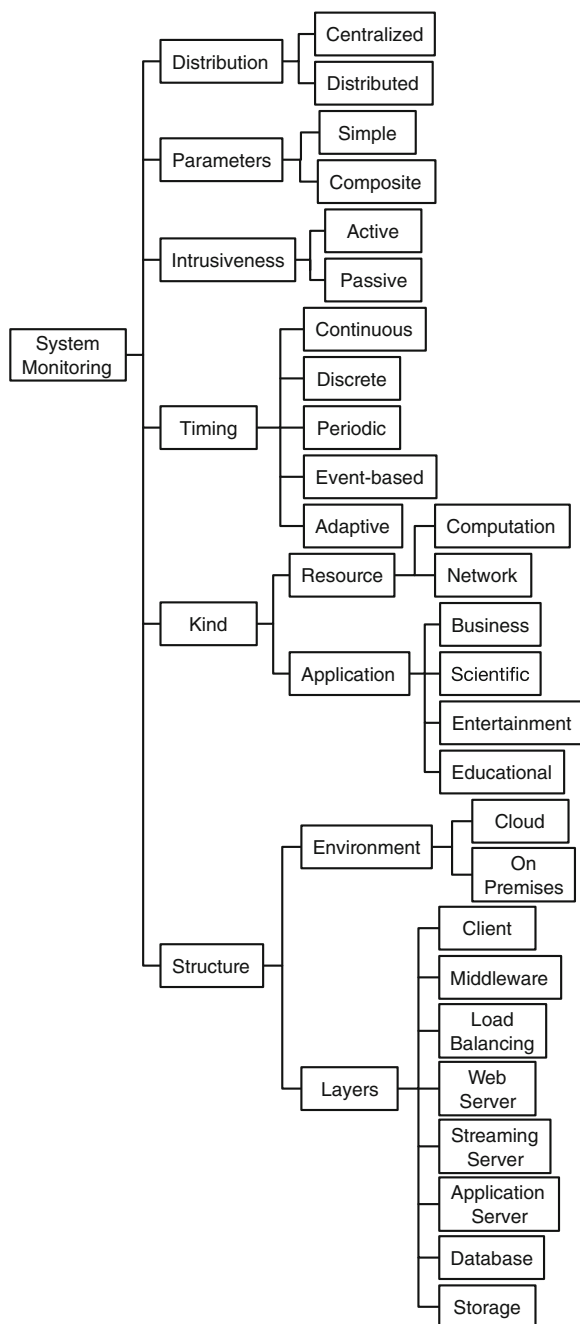


Fig. 2 Main dimensions of monitoring systems based on the literature

for consumers in operational areas. The data collected through monitoring plays a central role in representing the system state and enabling enhanced resource management decisions to be made either in an automated or manual way [64]. In addition, surveying the properties of resource and application monitoring can identify the fitness of monitoring tools in serving specific conditions.

After a thoughtful analysis of the surveyed initiatives, we were able to pinch the main dimensions which are discussed in such initiatives. Then, we performed classifications (using the defined dimensions) and analyze the result iteratively. Figure 2 presents the main dimensions of systems that are applicable for both resource and application monitoring. They are used to guide the discussion and categorize the state-of-the-art proposals, summarized in Table 2.

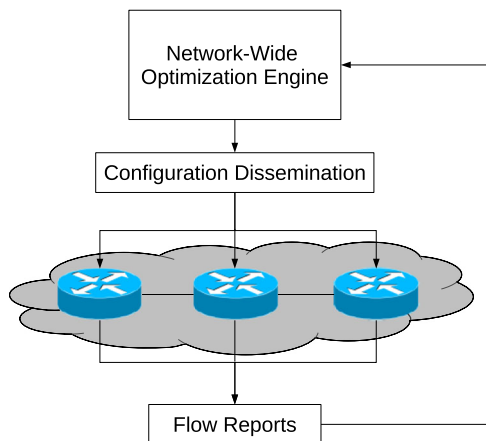
The architecture of the resource monitoring can be deployed using different distribution approaches. Despite not having a widely accepted taxonomy that defines and characterizes distribution aspects of network management models, works in monitoring usually consider distribution aspects which produce, at least, centralized and distributed models (e.g., [50]). These models are effectively accepted in the literature and present relevant characteristics considering the control of monitoring mechanisms investigated in this study (Fig. 3).

The parameters used to resource monitoring can be either single or composite. A single parameter refers to a specific monitoring target (i.e., CPU utilization). Most of the monitored resources (i.e., CPU, memory, disk) have a single-parameter metric to represent their current statuses, such as resource utilization or availability. On the other hand, composite parameters take a group of different metrics into consideration to provide a combined value. For illustration, the delay of a network is a metric that can be considered as a composition parameter of two single parameters: one-way and round-trip delays [66].

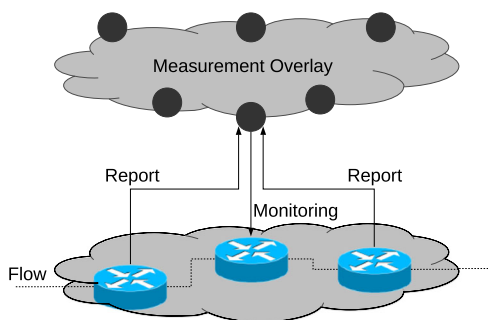
Application and resource monitoring can be performed through different levels of intrusiveness. The monitoring mechanisms can be classified as active or passive, considering the amount of disturbance they inject into the system [46]. Although both types share the same goal of monitoring, they have several differences that arise from the peculiarities of each kind of monitoring mechanism (e.g., the monitoring entities and their relationship). In an attempt to obtain the

Table 2 Classification of the monitoring proposals in literature

Proposal	Scope	Distribution	Parameters	Intrusiveness	Timing	Kind of resource or App	Environment
Farshchi et al. [20]	Resource	Centralized	Composite	Passive	Continuous	Computation	Cloud
Xu et al. [76]	Resource	Distributed	Single	Active	Adaptive	Computation	Cloud
Katsaros et al. [30]	Resource	Centralized	Composite	Passive	Continuous	Computation	Cloud
Sun et al. [69]	Resource	Distributed	Passive	Composite	Adaptive	Computation	Cloud
Borchert et al. [11]	Application	Centralized	Composite	Passive	Discrete	Business	On Premises
Röhl et al. [59]	Application	Centralized	Single	Active	Continuous	Scientific	On Premises
Carvalho et al. [15]	Application	Distributed	Composite	Active	Periodic	Business	Cloud
Al-Ayyoub et al. [5]	Application	Distributed	Composite	Active	Continuous	Business	Cloud
Balcas et al. [10]	Application	Distributed	Single	Active	Periodic	Scientific	On Premises
Mandal et al. [41]	Application	Centralized	Single	Active	Event-based	Scientific	On Premises
Khan et al. [31]	Application	Centralized	Single	Active	Continuous	Entertainment	Cloud
Fittkau and Hasselbring [22]	Application	Distributed	Single	Active	Continuous	Business	Cloud
Aaziz et al. [1]	Application	Centralized	Composite	Passive	Continuous	Business	On Premises
Tonouchi [70]	Application	Centralized	Single	Active	Event-based	Business	On Premises



(a) Centralized Model. Adapted from Sekar et al [64].



(b) Distributed Model. Adapted from di Pietro et al [52].

Fig. 3 Control distribution of monitoring mechanisms model

best from both worlds, some control approaches aim at integrating active and passive mechanisms toward specific reasons and goals.

Pervasive monitoring can facilitate the behavior of computing systems. However, such monitoring can increase the cost and overhead prohibitively. Thus, it is necessary to find a trade-off to monitor the system optimally (if possible). In this context, there are five major timing approaches for monitoring:

- **Continuous.** The system keeps a continuous monitoring over time;
- **Discrete.** The monitoring occurs only in specific moments;
- **Periodic.** It is possible to describe the monitoring times using a mathematical function;
- **Event-based.** Monitoring is triggered by the detection of an event;
- **Adaptive.** The monitoring system can adapt to environmental conditions.

All of these approaches can be evaluated using statistical measures. For example, it is possible to use a composition of a measure of the central tendency (e.g., mean) and a measure of spread (e.g., standard deviation) as chosen metrics. Finally, it is also possible to characterize the monitoring data as temporal series.

In the following subsections, we select state-of-the-art monitoring initiatives and classify them based on review questions to produce an integrated perspective of them. First, we present initiatives regarding

resource monitoring. After that, we discuss those focused on application monitoring.

3.1 Resource Monitoring

The monitoring of resource parameters is a functionality usually provided by computing systems. Such parameters can be tracked either in physical or virtualized environments. For example, CPU utilization, network delay, and packet losses. These parameters can assist computing infrastructure providers to maintain their resources operating at high efficiency and also to detect abnormal usage variations. We consider that resource monitoring occurs when there is no connection to application information. In this subsection, we present the evaluation dimensions for resource monitoring employed in the present study.

A computing system can monitor several resources, divided into two main dimensions: computation and network. Computation-based tests are related to monitoring activities aimed at gaining knowledge about and inferring the status of real or virtualized resources. Some examples are server throughput, CPU Speed, CPU time per execution, CPU utilization, memory page exchanges per execution, disk/memory throughput, throughput/delay of message passing between processes, response time, VM startup, acquisition or release time, and up-time. Network-based tests are related to the monitoring of network-layer metrics. This set includes one-way and two-way delays, jitter, throughput, packet loss, available bandwidth, capacity, and traffic volume [2].

The environment where the resources are deployed can be used to classify the surveyed initiatives. On the one hand, the deployment can be performed as a cloud. Thus, the user consumes the monitored resources as a service considering the different well-known models (e.g., Infrastructure-as-a-Service and Platform-as-a-Service). On the other hand, the resources can be deployed on premises. In this context, we consider that the user has an internal infrastructure, which provides the processing of their computation tasks. The surveyed initiatives represent the zeitgeist of the resource monitoring. These initiatives are depicted in Table 2.

Farshchi et al. [20] propose a monitoring approach to detect errors in the execution of DevOps operations, particularly for rolling upgrade operations. Some operations analyzed are backup, redeployment, upgrade, customized scaling, and migration.

The authors develop a regression-based approach that leverages the correlation between operations activity logs and the effect of operation activities on cloud resources. Such approach is composed of metric selection based on regression analysis and the use of the output of a regression model of selected metrics to derive assertion specifications.

Xu et al. [76] propose a distributed collaborative monitoring model (DCMM) for cloud computing infrastructures. This model provides self-organized capabilities based on mutual perception and balanced monitoring of each node. In addition, an adaptive threshold control algorithm (ATCA) is proposed to dynamically adapt the sets of thresholds used for notification purposes to identify unnecessary duplicate information sent back to the monitoring tool. ATCA is based on historical monitoring records. Furthermore, the model contributes to real-time monitoring and data consistency within the monitoring architecture.

Katsaros et al. [30] propose a service framework that allows the cloud infrastructure to monitor the energy consumption. The measured value is used to calculate its energy efficiency and evaluate the gathered data to put in place an effective Virtual Machine (VM) management. The authors claim that providers can address the energy-efficient management of their resources and running services using the proposed framework. The monitoring infrastructure can provide real-time and predicted information about applications, resources, and energy consumption. The assessment of such metrics is used to drive self-management policies to fulfill the provider's energy efficiency requirements.

Sun et al. [69] propose the Bionic Autonomic Nervous System (BANS), a resource monitoring model that deals with the typical issues of scalability and flexibility of monitoring systems. Besides, BANS also support autonomy, which provides an efficient design that enables the monitoring system to disperse pressure of primary monitoring node on data processing and reduce the network traffic load. The system was deployed in the Collaborative Autonomic Computing Laboratory cloud platform.

Several initiatives on resource monitoring were presented in this subsection. Despite some difference, physical and virtualized environments require a control on the employed resources. Consequently, it is necessary to collect the information about these resources. Besides that, different kinds of resources

have subtle differences in their monitoring as well as the employment in cloudified environments.

3.2 Application Monitoring

Application monitoring is an established process to measure the performance of different kinds of applications. This monitoring type is essential in several scenarios since it can provide data to management tools to improve the efficiency of computing jobs and hardware. One of the main goals of the application monitoring is to extract application patterns. Such patterns can be used to provide performance feedback as well as to offer information on the optimization space of specific applications. In this subsection, we present the evaluation dimensions for application monitoring employed in the present work.

Different kinds of applications can be monitored. To guarantee that the service-level objectives for the application are satisfied, it is first necessary to monitor such applications. In this context, some applications require a complete awareness of their running state. In the present research, it is considered the software components which run on top of the operating system. Some examples of these kinds are business applications, scientific applications, entertainment applications, and educational applications.

Several applications (e.g., business applications) expose an intricate multi-layered layered structure [55]. These layers usually are composed of clients, middlewares, load balancers, web servers, streaming servers, application servers, indexing server, database systems, and storage services. In addition, the layers can be either allocated to different hardware resources or distributed across cloud layers including Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Monitoring approaches are mostly oriented to perform jobs on a unique of the aforementioned layers. Hence, the challenge here is to develop monitoring tools that can collect performance parameters of application components across n layers.

Borchert et al. [11] propose the integration of non-intrusive application monitoring of response times and subjective user ratings on the perceived application performance. In this context, the authors evaluated the possibility of predicting the Quality of Experience (QoE) using different machine learning approaches. The performed evaluation implied a high correlation

for specific users, but the authors stated that was not possible to derive a generic model due to the high complexity of enterprise systems and the disturbances of the day-to-day business by user studies.

Röhl et al. [59] propose the *LIKWID monitoring stack*, a job-specific performance monitoring system using Hardware Performance Monitoring (HPM) data, system metrics, and application-level data for small to medium sized commodity clusters. The proposed framework is built on top of the LIKWID tools library in a modular fashion. In addition, *LIKWID* is designed to integrate into existing monitoring infrastructures to speed up the change from pure system monitoring to job-aware monitoring.

Carvalho et al. [15] propose a security assurance platform that allows monitoring the multi-cloud application deployed in different Cloud Server Providers (CSPs). This platform detects potential deviations from security Server Level Agreements (SLAs) and triggers countermeasures to enforce security during application runtime. Some monitoring tasks that can be performed are data capture, filtering and storage, events extraction and statistics collection, and traffic analysis and reporting. Besides that, the monitoring agent can correlate network events to detect performance, operational, and security incidents. The platform is composed of several monitoring agents. One of its agents is the application monitoring agent which is responsible for extracting the information from the application environment.

Al-Ayyoub et al. [5] focus on improving the resource utilization by optimizing the resource provisioning, which leads to reduced cost, improved customers experience, and shortened completion time. The authors employ a multi-agent framework in which different agents are responsible for different tasks including the monitoring of customers and available resources as well as the provisioning of resources based on customers requests. Finally, the authors introduce the concept of *TaskFlow* which improves the elastic provisioning of the customer resources.

Balcas et al. [10] propose *MonALISA* (Monitoring Agents using a Large Integrated Services Architecture), which was developed with the support of the software and computing program of the CMS and ALICE experiments at the Large Hadron Collider (LHC). The framework is composed of autonomous, multithreaded, self-describing agents. Such agents can collaborate and cooperate to perform data gathering

and processing. The information analysis and processing performed by these agents can be done in a distributed way. MonALISA provides the ability to add to the system increasing degrees of intelligence and to reduce the complexity of monitoring tasks.

Mandal et al. [41] present a prototype of an end-to-end system for modeling and diagnosing the runtime performance of complex scientific workflows. This prototype is integrated with the Pegasus workflow management system and Aspen performance modeling. The integrated framework improves the understanding of complex scientific applications on large scale complex infrastructure and detects anomalies and supports adaptivity. In addition, the authors present a black box modeling tool, a comprehensive online monitoring system, and anomaly detection algorithms.

Khan et al. [31] propose an approach for the monitoring of High-Availability (HA) applications at the service level, its integration using the SA Forum middleware, and an elasticity engine for triggering resource provisioning on the application level. The approach allows the monitoring of application processes in the traditional manner and the mapping of this workload to their combined service level workload. Resource usages are aggregated and mapped to the service level workload using a distributed client-server architecture.

Fittkau and Hasselbring [22] present an application-level monitoring using cloud computing for large software landscapes comprised of several hundreds of applications. The proposed monitoring is elastic and distributed, enabling it to add or remove worker levels dynamically. This approach prevents overloading the analysis master application without interrupting or pausing the actual live analysis of the monitored data. The proposal addresses the potential bottleneck of analysis applications when monitoring several programs, which may result in reduced monitoring quality and coverage.

Aaziz et al. [1] propose PROMON which is a framework for application monitoring in the production environment, but its design concentrates on the front end issues of offering easy to use application instrumentation. The authors propose the integration of PROMON with LDMS, a proven efficient HPC system monitoring framework. In addition, they depict a case study in integrating these instrumentation and monitoring models (i.e., PROMON and LDMS) to apply to HPC monitoring issues.

Tonouchi [70] introduces a prototype of an application monitoring system, which enables the operator team to monitor the method calls in Java application programs. This prototype can also localize a failure method with statistical debugging. Since the internal behavior of application programs is a black box to the operator team, the author aims at easing the monitor of the internal behavior of running applications for an operator team. Furthermore, the prototype helps development teams to revise the applications.

3.3 Analysis of Monitoring Methods

Several initiatives on application monitoring were presented in this subsection. Such monitoring provides valuable information into running applications. In this context, it is possible to divide the methods employed in application monitoring in Data Analysis methods and Instrumentation methods. Regarding Data Analysis, the surveyed initiatives can be classified in *Machine Learning*, *Artificial Intelligence*, and *Anomaly Detection*. On the other hand, considering the Instrumentation, the initiatives can be classified as *Black Box* and *Code Annotation*. A summary of the advantages and disadvantages of these methods is presented on Table 3.

4 Resource Management Techniques

Computing infrastructures provision computational nodes with a set of resources (e.g., CPU, memory, storage, bandwidth) for users to perform their tasks at a reasonable price, typically those tasks that would be infeasible or too costly otherwise [25]. Some examples of computing infrastructure organizations are clusters, grids, data centers, and clouds. There are two actors with different goals regarding these infrastructures: providers and clients [17]. Providers manage and provision the resources of the infrastructure, aiming at maximizing resource utilization, to increase the number of clients they can provide, while reducing the energy consumption, which is the most significant expense in the infrastructure. Clients use the offered resources and services, attempting to obtain the best performance (e.g., lowest execution time, highest availability) with the lowest cost.

Independent of the goal, the resources are limited and need to be managed efficiently to achieve them

Table 3 Methods of the monitoring proposals in literature

Class	Methods	Advantages	Disadvantages
Data Analysis	Machine Learning	Data scalability	Centralization
	Artificial Intelligence	Adaptability	Complexity
	Anomaly Detection	No need for signatures	False positives
Instrumentation	Black Box	No need for application modifications	Less granularity
	Code Annotation	Accuracy and traceability	Need for application modifications

[34, 40, 47]. We discuss three techniques for improved resource management of computational infrastructures: cloud elasticity, load balancing, and resource scheduling. For each technique, we present its oper-

ation, goals, approaches, and a set of state-of-the-art research papers. Each technique is discussed individually, but in practice, they work together to improve the resource management. We begin discussing cloud

Table 4 Summary of the proposals of resource management

Proposal	Technique	Approach	Properties	Metrics
AutoElastic [54]	Cloud Elasticity	Horizontal and reactive scaling	Automatic, transparent, and asynchronous operation	Execution time, CPU load, and Cost
Fuzzy-PID [51]	Cloud Elasticity	Horizontal and reactive scaling	Decisions based on the combination of CPU and network load	Stability and Robustness
MEC [63]	Cloud Elasticity	Vertical scaling	Adjusts memory and storage capacity of VMs dynamically	Execution time and throughput
ElasticCloud [12]	Cloud Elasticity	Horizontal scaling reactive and predictive	Reactive upscale and predictive downscale of resources for a web application	CPU load, CPU bottlenecks, number of scaling operations, number of downscaling mistakes
Ma et al. [39]	Load Balancing	Container migration and dynamic CPU allocation	Two-level resource coordinator: host (dynamic CPU allocation) and cluster (container migration)	Execution time and load imbalance
HAVEN [53]	Load Balancing	Congestion monitoring and horizontal scaling	Distributed, flexible, dynamic, and adaptive packet processing with SDN controller	Average reply time and average reply rate
RIAL [67]	Load Balancing	VM migration and resource weights	Assigns dynamic weights to resources according to their use intensity for a better system snapshot	Number of VM migrations, VM performance degradation, and VM communication cost
MDTS [4]	Resource Scheduling	List-based task scheduling	For ranking, uses the Median Absolution Deviation of the Expected Time to Compute of tasks	Makespan speedup, efficiency
Duan et al. [19]	Resource Scheduling	Multi-objective scheduling of bags-of-tasks	Applies a game theoretical approach using a sequential cooperative game	Scheduler execution time, makespan, cost, system-level allocation efficiency, fairness
Ghaderi [26]	Resource Scheduling	Randomized algorithm for VM placement	Employs a set of queues with dedicated clocks ticking at a Poisson distribution	Average delay and throughput

elasticity [6, 24, 48], the property of cloud computing that enables resource scaling to meet the dynamic user demands. Then, we review load balancing [49, 72], which is applied to the several types of resources available (e.g., CPU, memory, network, storage). Finally, we present resource scheduling [16, 34, 40], focusing on the initial resource allocation in the computing infrastructures. Table 4 summarizes the content presented in this section and guides the discussion.

4.1 Cloud Elasticity

Elasticity is one of the five fundamental characteristics of the cloud computing model, together with on-demand self-service, broad network access, resource pooling, and measured service [44]. The National Institute of Standards and Technology (NIST) defines elasticity as the competence to add and remove resources for upscaling or downscaling them corresponding to the load demand [44]. From the consumer perspective, elasticity results in “unlimited” resources that they can allocate at any given moment. It is important to mention that the applications may support dynamic scaling in order to exploit elasticity features.

Figure 4 presents a classification of cloud elasticity proposals [24] with four properties: scope, method, policy, and purpose. The *scope* defines where is the control and execution of the elasticity: infrastructure, platform or applications. In all cases, the infrastructure supports virtualization, using either Virtual Machines (VMs) or containers. The *method* specifies how the resources scale: replication, resizing or migration. The *policy* sets when the elasticity is triggered: manually or automatically. In the latter case, it can be either reactive or predictive. Lastly, the *purpose* describes the primary goal of the elasticity proposal: increase performance or utilization, or decrease cost or energy.

Cloud computing infrastructures use virtualization to manage their resources. This approach enables the cloud infrastructure to share physical resources among different tenants and to have a fine-grained control of the resources, which helps to provide elasticity for the executing applications. There are two main virtualization techniques that clouds can employ: VMs [12, 51, 54, 63] and containers [48]. VM is the most used virtualization technique in the cloud. Each VM instance is a full OS image with the binaries and libraries required for its execution. Clouds use VMs to allocate resources to users and allow them to execute

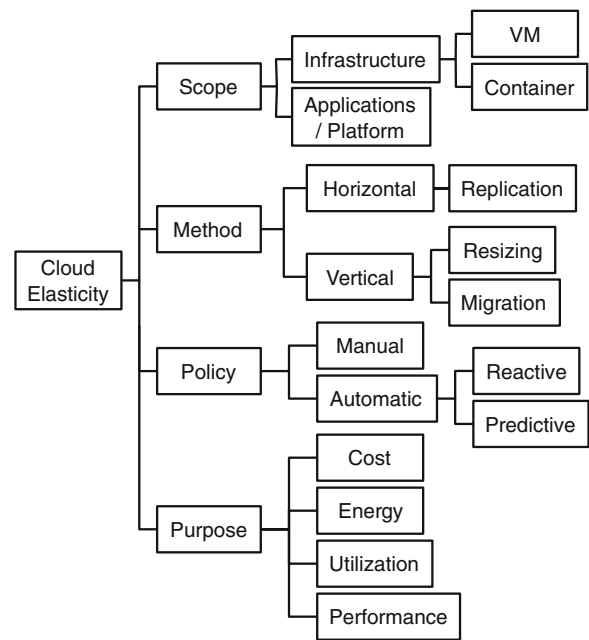


Fig. 4 Classification of cloud elasticity solutions adapted from [24]

their applications. Through isolation, multiple VMs may share the resources of a single physical node and overcome security concerns. VMs have two main downsides: the image size may be significant, requiring more memory and storage from the physical node, and the booting process may take minutes, reducing the application performance [48].

In an attempt to overcome these two limitations, some techniques can be employed. Still considering VMs, it is possible to anticipate the provisioning of VMs to mitigate the provision lag (“pre-provisioning”) Besides that, clouds are starting to consider containers for virtualization. Instead of being an OS image, they are self-contained software packages, ready to be executed as a regular OS process. Consequently, they require less memory, storage, and start-up time [48]. Despite addressing the VM limitations, they have shortcomings regarding managing multiple container instances, dependencies for executing applications (not self-contained), and security (especially data and network management). Also, containers have a limitation on the kernel level, since the instances should be able to use the same kernel as the others, otherwise you can’t move them around.

The cloud infrastructure can offer elasticity through horizontal scaling, vertical scaling or a hybrid

approach that combines both [6, 24, 25, 47]. The horizontal scaling is the most popular approach and widely deployed in current solutions [12, 51, 53, 54]. It relies on replicating and deleting VM instances to adjust the allocated resources according to the load demand. The vertical scaling, in its turn, is based on resource resizing at runtime [63]. Instead of creating new replicas, it allocates or deallocates resources to the running VMs. Optionally, the cloud can also migrate VMs to a more capable computational node if there is not enough resource in the current one. Regardless of the approach, the cloud infrastructure usually couples the elasticity with load balancing for more efficient use of resources [6].

The cloud decides when to perform its elasticity operations either manually or automatically. Using the manual approach, either the administrator or the client needs to allocate or deallocate the resource. An automatic approach enables resources to be added or removed without human intervention. If it is automatic, it can be further divided into reactively or predictively [6, 24]. The reactive approach uses a set of rules configured by the user that defines when an upscale or downscale should occur [12, 51, 54, 63]. On the downside, it requires a priori knowledge about the application and its load demand. The alternative is to

use a predictive approach that attempts to forecast the user demands and adjust to them before it forms a bottleneck [12]. The main techniques for load prediction will be presented in Section 5.

We selected a set of state-of-the-art proposals that cover a variety of different alternatives regarding properties of cloud elasticity solutions. They are summarized in Table 4 and discussed next.

AutoElastic [54] proposes an elasticity model for High-Performance Computing (HPC) based on horizontal and reactive scaling, exemplified in Fig. 5. The elasticity offered by AutoElastic is automatic, transparent, and asynchronous. Being automatic means that users do not require to configure the elasticity rules for scaling. Instead, AutoElastic monitors the resource consumption of the application, setting adjustable rules to meet the demand. Transparency means that AutoElastic includes tools for transforming a parallel application into an elastic one without user intervention. The replication is made asynchronous, which allows the application to continue executing while replicating VMs. The results of the evaluation showed that AutoElastic improves the execution time of applications by up to 26 percent.

Fuzzy-PID [51] proposes a horizontal and reactive elasticity solution that makes decisions based on the

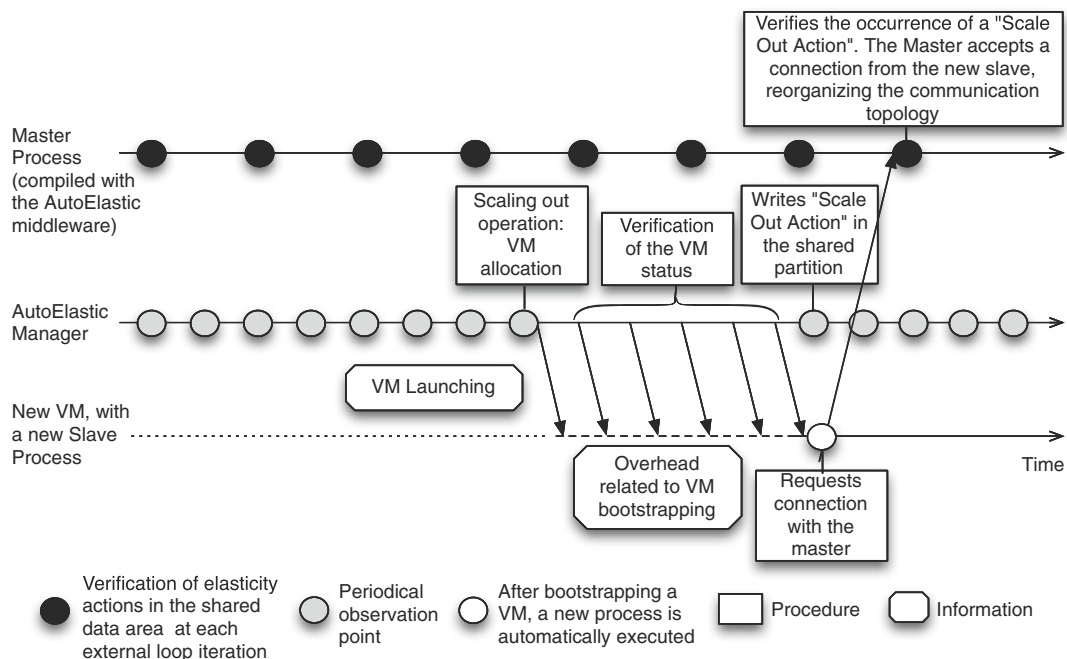


Fig. 5 Exemplification of cloud elasticity using AutoElastic. Adapted from Galante et al. [25]

combination of CPU and network load. It is composed of four components: monitoring, fitness, control, actuation. First, Fuzzy-PID monitors metrics of CPU and network load from the nodes. Then, it combines them in a single load index and builds a state of the cloud system. Next, it calculates the adjustment of the VMs allocated for single-loop applications. Finally, Fuzzy-PID performs the changes, adding and removing VMs. The evaluation shows that the self-adaption of Fuzzy-PID results in a robust, flexible solution that performs better than other similar proposals.

MEC [63] proposes a vertical elasticity solution to adjust memory or storage capacity of VMs according to their demands because of the increased influence of these resources on performance. The framework developed manages the memory allocation and VM states dynamically and independently of hypervisor or application. It uses a two-level hierarchy composed of the virtual and physical machine layers. This multilevel organization enables MEC to manage the resources more efficiently by dividing the responsibilities. The cloud level addresses VM allocation, VM migration, and load balancing. The physical layer handles resource sharing of VM instances. The virtual layer deals with the application performance.

ElasticCloud [12] proposes a horizontal elasticity mechanism that employs both reactive and predictive techniques to upscale and downscale resources for a web application in the cloud. The proposal uses reactive upscaling to add new VMs and predictive downscaling to optimize the resource use. It is composed

of two components: Elasticity Interface to manage resource provisioning and Load Balancer to forward requests to the VMs. The results show that Elastic-Cloud improves performance and stability compared to other proposals by decreasing the number of CPU bottlenecks and mistakes in scaling operations.

4.2 Load Balancing

In computing infrastructures, the load (e.g., memory, CPU, bandwidth, and storage) is highly dynamic and heterogeneous [77]. As a consequence, it may cause an imbalance of load distribution among the available resources over time. Load balancing is the process that manages the load distribution in the system in a way that resources do not have a high load (bottleneck) or a low load (waste of resources) [35, 38]. The overall goal of load balancing is twofold: from the infrastructure perspective, use resources efficiently as well as maintain system stability and, from the user point-of-view, improve application performance [49, 56, 72, 73, 77]. Computing infrastructures perform load balancing by moving and migrating VMs, applications or processes from their current node to another one. Figure 6 exemplifies the cases without load balancing (a) and with load balancing (b), evidencing the load difference among the processes when comparing both.

Figure 7 presents a classification of load balancing proposals [49]. At the highest level, load balancers are either static or dynamic depending on whether they have full a priori information of the load and

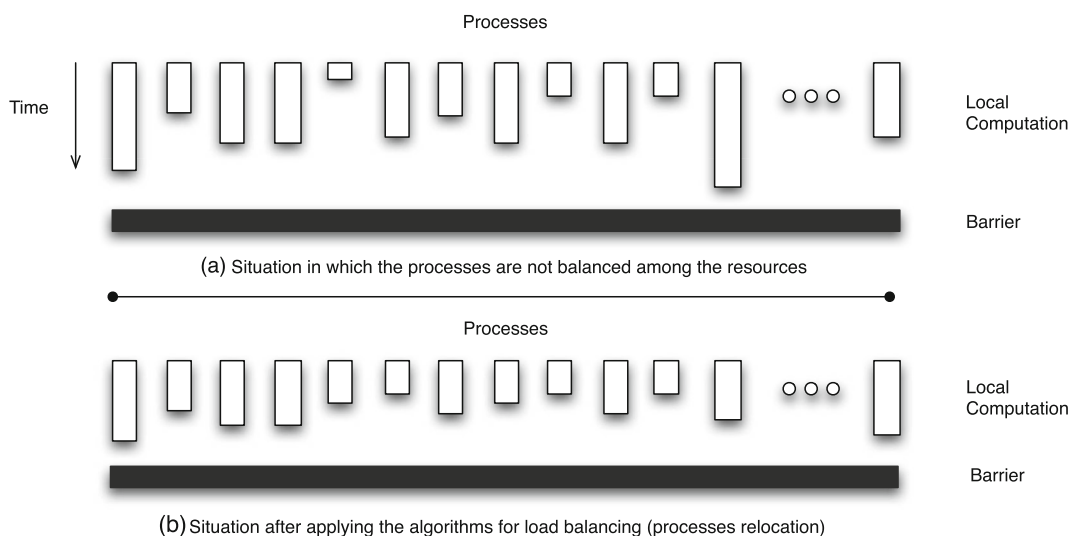
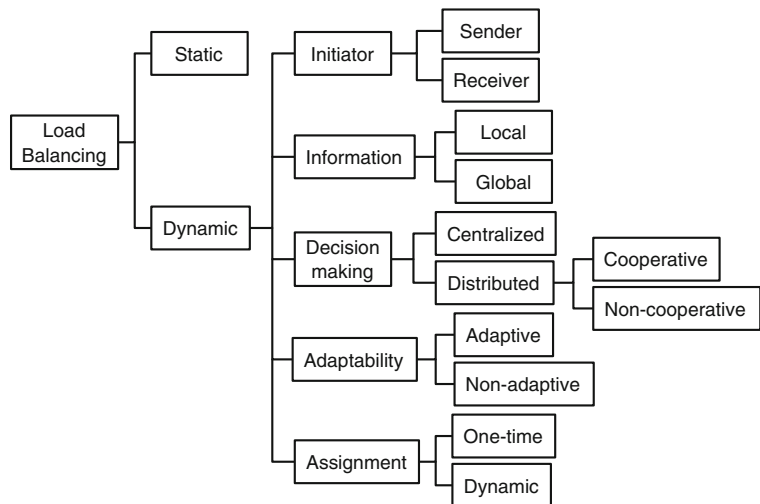


Fig. 6 Exemplification of load balancing. Adapted from Galante et al. [60]

Fig. 7 Classification of load balancing solutions. Adapted from da Rosa Righi et al. [49]



resources available (static) or not (dynamic). In computing infrastructures, dynamic load balancing is the vast majority of the cases and has five properties: initiation, information, decision making, adaptability, and assignment. The *initiation* defines who triggers a load balance: the overloaded (sender-initiated) or the underloaded node (receiver-initiated). The *information* configures if the load balancing is performed using local information from each node and neighbors or global information from the entire system, which affects communication cost and scalability. The *decision making* sets if the control logic of the load balancer is centralized or distributed. If distributed, it can also be cooperative or non-cooperative between nodes. The *adaptability* refers to whether the load balance adjusts its parameters to the changes seen in the network or not, but it may degrade the system due to misconfiguration. Lastly, the *assignment* defines if the load migrates multiple times or only one time, to avoid performance degradation.

Load balancing in computing infrastructures is most often dynamic than static given the load heterogeneity and frequent changes in the computing infrastructure caused by multiple tenants [77]. The most significant difference between them is data knowledge. While in the static load balancing, the information about the entire system state and load characteristics is known a priori, in the dynamic, it has to be collected and updated during execution. Therefore, the system has to monitor the resources, as presented in Section 3 to detect a load imbalance. Once it found one, it computes how to balance the load, calculate the cost, and

determine if the load balancing benefits surpass its cost [56].

Regarding initiation, the load balancing proposals have two alternatives: sender-initiated or receiver-initiated policy [49, 72]. Using a sender-initiated policy, overloaded nodes look to move part of its load to underloaded ones [39, 53, 67]. Conversely, the receiver-initiated policy defines that underloaded nodes try to find overloaded ones to receive part of their load. The best policy depends on the probability of finding a matching node to transfer load, which is represented by the average load of the system [72]. For example, sender-initiated policies tend to work better on systems with moderate load because it is easier to find an underloaded node. Similarly, receiver-initiated policies perform better on systems with a heavy average load because of the high probability of finding overloaded nodes.

The decisions of the load balancer can be done in a centralized or distributed fashion [49, 72]. In a centralized design, one node is responsible for the load balancing decisions [39, 53, 67]. The other nodes in the system send their information to the load balancer. With the information about the state of the entire system, it can calculate the best load distribution using even optimal algorithms. Despite being easier to implement and potentially providing optimal solutions, it may become overwhelmed, reducing the system scalability, fault tolerance, and performance. The decentralized design aims at improving the load balancing reliability at the cost of the solution quality [52]. All the nodes are involved in the load balancing

decisions. They exchange information between them and may attempt to find a common solution (cooperative) or execute the load balancing autonomously (non-cooperative).

The adaptability of a load balancer refers to its capacity to adjust over time because of changes in the environment [49]. An adaptive load balancer may change its scheduling policies or parameters of the algorithm (i.e., thresholds) due to observable patterns or previous decisions [39, 53, 67]. This property fits better a system that has significantly different utilization patterns, enabling the load balancer to adapt and operate accordingly. A non-adaptive load balancer uses static parameters and configurations and suits best for systems with constant utilization patterns. Load balancers may assign VMs a single time or multiple times when operating [49]. In other words, when assigning only one time, the load balancer focuses on distributing the load in the initial allocation and does not migrate that VM until it stops executing. The upside is avoiding multiple migrations of the same VM, which can degrade performance. On the downside, it may not achieve the optimal load balance due to the VM immobility. The multiple-times assignment allows load balancers to migrate VMs as many times as needed, providing the opposite results of the one-time assignment [39, 53, 67].

Next, we discuss a set of state-of-the-art proposals regarding dynamic load balancing. They all have similar properties, evidencing a current trend of this technique: sender-initiated, global, centralized, adaptive, and multiple-time assignment. The most significant difference between the proposals is the resources they balance: CPU, network bandwidth, and VMs. Table 4 summarizes the proposals detailed next.

Ma et al. [39] propose an adaptive resource management to balance the load of MPI programs using container virtualization. The proposal has three components: monitor, host resource coordinator, and cluster resource coordinator. The monitor collects execution information from containers. The host resource coordinator is responsible for managing resources within a host. It allocates CPU dynamically among the containers to reduce the load imbalance. The cluster resource coordinator operates at the inter-host level and addresses load imbalance between nodes by migrating containers between hosts. The evaluation demonstrates that the proposed solution can improve

the performance on average by 15%, reaching a maximum of 31% in some cases.

HAVEN [53] proposes a holistic load balancing and auto-scaling with horizontal elasticity for packet routing in multi-tenant cloud environments. It uses an SDN controller to execute the load balancing and auto-scaling decisions with four actions: monitor resources, calculate a load score, balance load, and scale. HAVEN collects statistics about the network bandwidth from the switches and resource utilization of VMs from their hypervisors. Based on this information, it scores each VM with a load index. When packets come in, two actions are performed. First, the controller selects the VM from the tenant with the lowest load and forward the packet for processing.

RIAL [67] proposes a Resource Intensity Aware Load balancing. The main feature of RIAL is dynamically assigning weights to resources according to their use intensity in a physical node. This approach results in a more precise snapshot of resource utilization and availability to allocate or migrate VMs in the system. The primary goal of RIAL is to reduce the time and cost to achieve load balance, to avoid future load imbalance and VM migrations that degrade performance, and to minimize the bandwidth cost by placing communicating VMs close to each other.

4.3 Resource Scheduling

Resource scheduling refers to allocating a set of user tasks to the set of resources available in the computing infrastructure, as shown in Fig. 8. Liu et al. [38] affirm that a scheduling method is the combination of policies, rules or algorithms to map the workload - jobs or tasks - to the available computing resources in order to achieve one or several objectives. The main problem is to find an efficient scheduler algorithm for resource management to optimize the performance metric of choice, deciding how and which resource allocate to

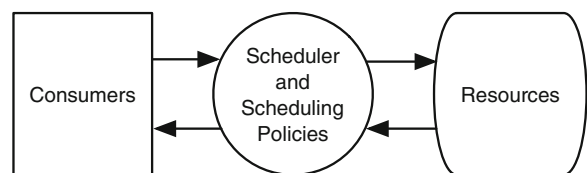


Fig. 8 Overview of resource scheduling. Adapted from Casavant and Kuhl [16]

run the corresponding task. Therefore, the scheduler algorithm is evaluated in two aspects: performance and efficiency [56]. The performance relates to the execution result using the allocation obtained by the scheduler. Typically, resource schedulers have the goal of minimizing the makespan, execution time, cost, among others [40]. The efficiency regards the complexity of the scheduler algorithm, which translates to its execution time to output the allocation.

Figure 9 presents a classical classification of resource scheduling proposals [16]. The structure is similar to load balancing (as depicted in Fig. 7) because they address close challenges. In the highest level, scheduling is classified according to the scope: local (single node) or global (computing infrastructure). The scope of this work is the global scope. Then, the global scheduling can be either static or dynamic. The static assumes a complete knowledge of the tasks to be executed and the system resources while the dynamic has little to no a priori information. Overall, resource scheduling has four approaches to organize the decision making: static, dynamic centralized, dynamic decentralized cooperative, and dynamic decentralized non-cooperative. Lastly, the resource scheduler can be optimal (e.g., enumerative, graph theory, math programming, queuing theory) or sub-optimal (e.g., approximate and heuristic algorithms). The specific implementation of the solution also depends on the decision-making structure which may enable or disable specific algorithms.

In a computing infrastructure, there are at least two scopes for scheduling: local and global [16, 23]. The local one refers to the OS scheduling of processes within a single node. The global scheduling is the allocation of user applications to the computing infrastructure. In this context, the system determines which VM instances are allocated to which nodes for execution. Typically, the resource scheduling or allocation considers some load balancing techniques to improve the resource management as well as the application makespan, execution time, cost, among other metrics [40]. For instance, it could attempt to allocate multiple VMs of the same application close to each other to minimize the communication cost.

Similarly to the load balancing, the global scheduling can be either static or dynamic [16]. The static one assumes a complete knowledge of the tasks to be executed and the system resources [71]. Having a priori knowledge about execution time, data size, communication pattern, relation dependency of tasks is a strong premise in computation infrastructures, mainly public clouds. If that was the case, the scheduler could calculate an optimal allocation of resources for the tasks. Given the complexity of tasks nowadays, obtaining an optimal scheduling solution even with all knowledge could prove to be too costly, requiring approximate or heuristic solutions [71].

The dynamic scheduling increases, even more, the complexity of allocating resources. In this case, the

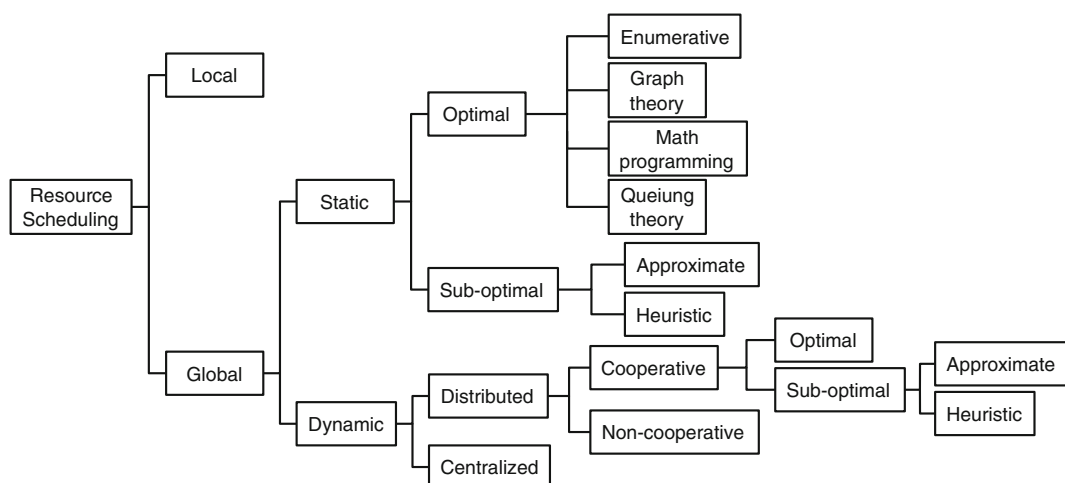


Fig. 9 Classification of resource scheduling solutions. Adapted from [16]

resource scheduler has little to no a priori information about the application behavior or the resource state in the infrastructure [72]. Further, several users share the computing infrastructure, sending new tasks over time, whose behaviors are unknown to the scheduler. In this context, it is infeasible to find the optimal scheduling solution and even sub-optimal ones can be costly. Therefore, computing infrastructures rely on resource scheduling and load balancing afterward to manage the resource and use them efficiently.

We choose a set of state-of-the-art works about dynamic resource scheduling, each using a different algorithmic to address the problem. Table 4 presents the summarized information of the proposals discussed next.

MDTS [4] proposes a novel list-based task scheduling based on the Median Absolute Deviation of the Expected Time to Compute of tasks for cloud computing. The proposal also uses coefficient-of-variation to take into account task and node heterogeneity when calculating the computational and communication costs. After ranking the tasks, MDTS schedules them in the available VMs. Iteratively, it selects the highest ranked task and calculates the earliest startup time and earliest finish time of the combination of each task and VM. The VM with the lowest finish time is selected for the allocation using an insertion-based policy considering the data dependencies for the execution. The evaluation indicates that the MDTS algorithm improves the makespan, speed up, and efficiency compared to other proposals.

Duan et al. [19] propose a multi-objective scheduling algorithm based on a game theoretical approach for the execution of bags-of-tasks overflows in clouds. It is formulated as a sequential cooperative game composed of application managers (players) that attempts to optimize the execution time and economic cost of a bag-of-tasks while satisfying network bandwidth and storage constraints (strategies). A game-multi-objective algorithm implements the scheduling algorithm using three phases: initial distribution of tasks, game stages, and player elimination. The second phase is played repetitively trying to optimize the scheduling obtained by each player making new decisions based on the preceding ones. Once a player completes the jobs, it is eliminated and release the resources. The evaluation shows that the solution outperforms similar proposals regarding makespan, cost,

system-level efficiency and fairness with faster convergence time.

Ghaderi [26] proposes a class of randomized algorithms for placing VMs in servers to achieve maximum throughput without VM migration. The system is modeled as a set of servers and queues of VMs containing jobs that arrive over time. Both the servers and VMs have a set of different resources, such as CPU, memory, and storage, that has to be considered for the scheduling. Once a VM is allocated, it executes until it finishes, when the VM releases the resources consumed. The proposal addresses the problem by having queues with a clock that ticks following a different Poisson distribution for each type of job. Every time a clock ticks, the algorithm tries to fit a job of that queue in a server. This approach has low complexity, is scalable, and requires no coordination between servers. The evaluation presents delay performance results comparable to heuristics.

4.4 Analysis of Resource Management Methods

Three methods of Resource Management Techniques have been presented in this section. These techniques for improving resource management have been, cloud elasticity, load balancing, and resource scheduling. For each technique, it was presented its objectives and approaches. In addition, they have been individually discussed and specified, but they work together to improve resource management. A summary of the advantages and disadvantages of these methods is presented on Table 5.

5 Load Prediction

In this study, load prediction refers to forecasting of resource consumption within a computing infrastructure. This process enables proactive management actions to be executed before a predicted resource depletion occurs, in an attempt to avoid it [27, 80]. An overloaded state is very detrimental for computing infrastructures and executing applications, because of its negative effects on performance, such as high response time, connection timeouts or even denial-of-service.

The load prediction process comprises two steps: resource monitoring and data analysis. The first one

Table 5 Classification of the monitoring proposals in literature

Method	Advantages	Disadvantages
Cloud Elasticity	On-demand computing, common coding, reactive and predictive of resources for web, asynchronous operation and ease of implementation	Downtime, security and privacy, vulnerability to attack and limited control and flexibility
Load Balancing	Load-based balancing strategies that monitor usage levels on individual computers and easier deployment	Additional configuration is required and hardware-based load balances cost more
Resource Scheduling	Organization of the user's task set and the scheduling method that maps the workload	Main problem is to find an efficient scheduler algorithm for resource management to optimize the performance metric

regards the collection of metrics and representation of the system state, as presented in Section III. The second step is the focus of this section and concerns analyzing data, creating pattern models, and predicting resource utilization. Based on the behavioral expectation of the data analysis, the system can act before an incident or system failure occurs.

For the analysis of a system over time, data is typically organized as time series, which is a set of observations representing sequential events that occurred in a given period. Time series can be classified as stochastic or deterministic processes [14]. The deterministic series has its behavior described by known mathematical equations, making it possible to predict a future value by merely calculating its formula. The stochastic series lacks a mathematical model to describe the behavior, either because it does

not exist or is unknown. Unfortunately, most patterns involving time series have a stochastic nature, which represents a challenge for load predictors to produce valuable outputs. Table 6 presents a collection of studies addressing load prediction, that are discussed in this subsection.

5.1 Linear Models

Deterministic processes can be represented by linear models, classified as stationary or non-stationary. A linear time series can be considered stationary when there is no tendency to increase or decrease its average value over time. That is, the data oscillate around a constant average and with a constant variance. A linear non-stationary time series is the opposite: its average value increases, decreases or has both tendencies over

Table 6 Summary of the proposals of load prediction

Proposal	Model	Approach	Properties
MA, AR and ARIMA [13, 14]	Linear	Moving Average, Auto Regressive and AR Integrated MA	Mean level, instant trend, no seasonality
Holt-Winters [28, 75]	Linear	Three parameters exponential smoothing	Mean level, instant trend and seasonality. Forecast one or more periods in the future
ANN [43]	Non-Linear	Back Propagation Artificial Neural Network	Model and prediction based on machine learning
ARIMA + ANN [79]	Hybrid	ARIMA and ANN	ARIMA to model linear part and residual from ARIMA to model nonlinear part
ARIMA + DWT [7, 18, 29, 32, 45]	Hybrid	ARIMA and Discrete Wavelet Transform	Low frequency from DWT applied on an ARIMA model and residual on an ANN model

time. The most known techniques used to analyze and predict linear time series are Moving Average (MA), Autoregressive (AR), AR Integrated MA (ARIMA) and Holt-Winters, and they are described next.

5.1.1 Moving Average (MA)

Moving average consists of calculating the simple arithmetic mean over a fixed-size subset of a time series. The values contained in this subset are permanently shifted as new data is added to the series. Thus, the moving average expresses the trend of the data in the time interval analyzed. As depicted in (1), given a sequence $\{a_i\}_{i=1}^N$, an n -moving average is a new sequence $\{s_i\}_{i=1}^{(N-n+1)}$ defined from the a_i by taking the arithmetic mean of subsequences of n terms.

$$s_i = \frac{1}{n} \sum_{j=i}^{i+n-1} a_j \quad (1)$$

For example, Fig. 10 shows an analysis of the number of earthquakes with magnitude greater than 7 on the Richter scale. The orange curve represents the number of events measured worldwide while the blue curve is the smoothed trending data, calculated using the moving average.

5.1.2 Autoregressive (AR)

An autoregressive (AR) model predicts future behavior based on past behavior. The model specifies that the output variable depends linearly on its previous values and a stochastic term known as white noise.

An $AR(p)$ model is an autoregressive model where specific lagged values are used as predictor variables. The results from one period affect following periods according to the value of p , which is called the order. The $AR(p)$ model is defined in (2), where $\varphi_1, \dots, \varphi_p$ are the parameters of the model, c is a constant, and ε_t is white noise.

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t \quad (2)$$

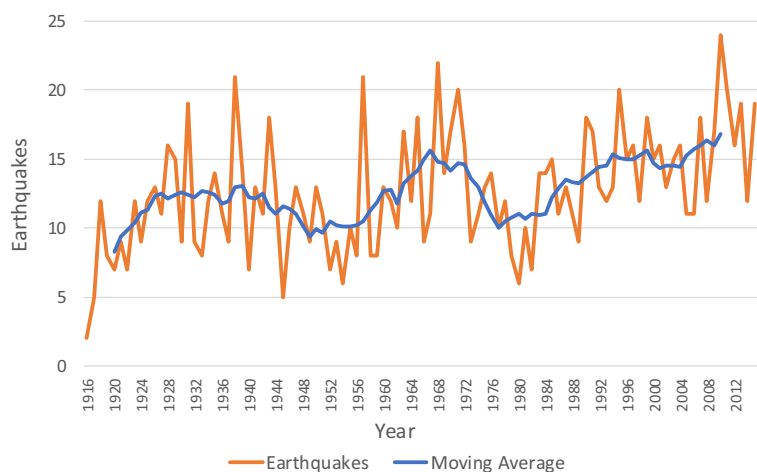
5.1.3 AR Integrated MA (ARIMA)

The ARIMA (Autoregressive Integrated Moving Average) models, introduced by Box and Jenkins [13], are the most popular and effective statistical models for time series forecasting. An ARIMA model combines three different processes: an autoregressive (AR) function, a moving average (MA) function regressed, and an integrated (I) part. They are based on the fundamental principle that future values of a time series are generated from a linear function of past observations, added to terms of white noise. That is, the underlying process that generates the time series has the form

$$y_t = \theta_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \dots - \theta_q \varepsilon_{t-q}, \quad (3)$$

where y_t and ε_t are the actual value and random error at time period t , respectively; ϕ_i ($i = 1, 2, \dots, p$) and θ_j ($j = 0, 1, 2, \dots, q$) are model parameters. p and q are integers and often referred to as orders of the

Fig. 10 Moving average



model. Random errors, ε_t , are assumed to be independently and identically distributed with a mean of zero and constant variance of σ^2 .

Equation (3) entails several important special cases of ARIMA family of models. If $q = 0$, then (3) becomes an AR model of order p . When $p = 0$, the model reduces to an MA model of order q . One central task of the ARIMA model building is to determine the appropriate model order (p, q). Box and Jenkins [13] propose to use autocorrelation function and the partial autocorrelation function of the sample data as the basic tool to identify the order of the ARIMA model.

5.1.4 Holt-Winters

The Holt-Winters forecasting procedure is a variant of exponential smoothing which is simple, works well in practice, and is particularly suitable for producing a short-term forecast. An exponentially weighted moving average is a mean of smoothing random fluctuations that have the following desirable properties: (i) declining weight is put on older data, (ii) it is extremely easy to compute, and (iii) minimum data is required.

A new value of the average is obtained simply by computing a weighted average of two variables, the value of the average from the last period and the current value of the variable. The exponential smoothing models are based on updating, for each period, up to three parameters:

- Mean level (simple smoothing model)
- Mean level and trend [28]
- Mean level, trend, and seasonality [75]

These models are also known in the literature as one, two, and three parameters exponential smoothing, respectively. Therefore, using the most comprehensive model, the Holt-Winters for trends and seasonality, in a period t , the pattern of demand (D_t) would be given by expression

$$D_t = (F + b_t)I_t + \varepsilon_t \text{ or } D_t = F + b_t + I_t + \varepsilon_t, \quad (4)$$

depending on whether we consider multiplicative or additive seasonality (I_t), respectively, where F denotes the mean level, and ε_t is the forecast error. The global updating in the multiplicative case is given by

$$F_t = \alpha \frac{D_t}{I_{t-p}} + (1 - \alpha)(F_{t-1} + b_{t-1}), \quad (5)$$

$$b_t = \beta(F_t - F_{t-1}) + (1 - \beta)b_{t-1}, \quad (6)$$

$$I_t = \gamma \frac{D_t}{F_t} + (1 - \gamma)I_{t-p}, \quad (7)$$

where D_t is the current demand; F_t , b_t , and I_t the mean level, trend, and seasonality predictions; α , β , and γ the smoothing parameters, usually constrained to (0, 1); and p is the number of seasons. The forecast for the subsequent period is $f_{t+1} = (F_t + b_t)I_{t+1-p}$. For forecasting more than one period into the future, the forecast is determined as follows:

$$f_{t+\tau} = (F_t + \tau b_t)I_{t+\tau-p}, \quad 1 \leq \tau \leq p \quad (8)$$

5.2 Nonlinear Models

Unlike linear processes that can be represented by a simple or compound mathematical equation, nonlinear processes - also known as stochastic or nondeterministic - are very difficult to be represented mathematically. Its unpredictability nature is often represented by complex computational structures, usually artificial neural networks.

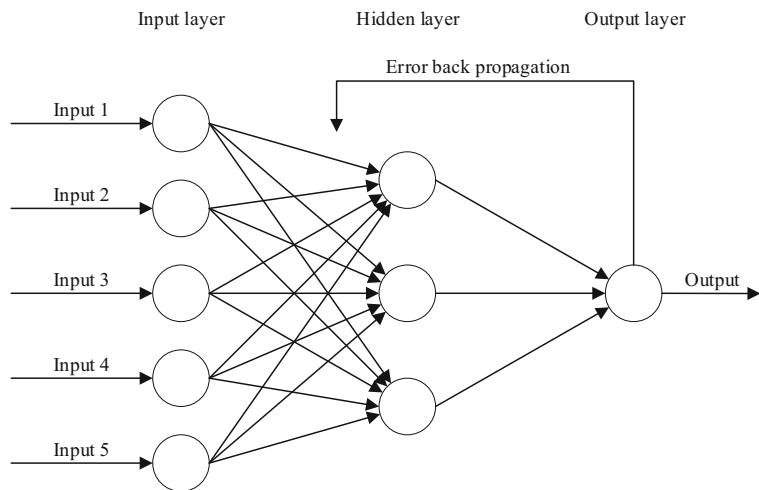
Artificial Neural Networks (ANN) are feasible computational structures used to model a wide range of nonlinear problems. A significant advantage of ANN models over other nonlinear models is that ANNs are universal approximations, which can estimate a large class of functions with a high degree of accuracy. No prior knowledge of the model is required during its construction process. Instead, the network model is mostly defined by the characteristics of the input data.

Single-layered feedback networks are the most widely used model for modeling and forecasting time series. A single hidden layer back propagation neural network consists of an input layer, a hidden layer, and an output layer, as shown in Fig. 11. Adjacent layers are connected by weights, which are always distributed between -1 and 1. Because a systematic theory to determine the number of input nodes and hidden layer nodes is unavailable, the most common means to determine the appropriate number of inputs and hidden nodes is via experiments or trial and error based on the minimum mean square error of the test data.

5.3 Hybrid Models

As shown earlier in this section, each technique obtains success to modeling data series according to

Fig. 11 Single layer artificial network with back propagation



its nature, linear or nonlinear. As a real-world event usually is not purely

linear or nonlinear, hybrid models aim to improve the accuracy of modeling and forecasting complex event-series. According to Khashei and Bijari [33], the motivation of the hybrid model comes from the following perspectives:

- i) It is often difficult in practice to determine whether a time series under study is generated from a linear or nonlinear underlying process or whether one particular method is more effective than the other in out-of-sample forecasting. Thus, it is challenging for forecasters to choose the right technique for their unique situations. Typically, different models are tried, and the one with the most accurate result is selected. However, the final selected model is not necessarily the best for future uses due to many potential influencing factors, such as sampling variation, model uncertainty, and structural change. By combining different methods, the problem of model selection can be eased with little extra effort.
- ii) Real-world time series are rarely pure linear or nonlinear. They often contain both linear and nonlinear patterns. If this is the case, neither ARIMA nor ANNs can be adequate in modeling and forecasting time series since the ARIMA model cannot deal with nonlinear relationships while the neural network model alone is not able to handle both linear and nonlinear patterns equally well. Hence, by combining ARIMA with

ANN models, complex autocorrelation structures in the data can be modeled more accurately.

- iii) It is almost universally agreed in the forecasting literature that no single method is best in every situation. The main reason is that a real-world problem is often complex in nature and any single model may not be able to capture different patterns equally well. Several empirical studies have already suggested that by combining several different models, forecasting accuracy can often be improved over an individual model. In addition, the combined model is more robust about the possible structural change in the data.

Following, we discuss two hybrid models well known in the literature, using ANN, ARIMA, and DWT.

5.3.1 ANN Model with Residual from ARIMA

According to Zhang [79], both ARIMA and ANN models succeed in their domains, respectively linear and nonlinear. However, none of them is a universal model suitable for all circumstances. On the one hand, the approximation of ARIMA models to complex nonlinear problems may not be adequate. On the other hand, the use of ANN to model linear problems produces mixed results. A study by Markham and Rakes [43] found that the performance of ANNs for linear regression problems depends on sample size and noise level. Therefore, it is not advisable to apply ANN blindly to any data. Since it is difficult to fully know the characteristics of the data in a real problem, the hybrid methodology that has both linear and nonlinear

modeling capabilities may be a good strategy for practical use. By combining different models, different aspects of the underlying patterns can be captured.

It is reasonable to consider that a time series $y_t = L_t + N_t$ can be composed of a linear autocorrelation structure and a nonlinear component, where L_t denotes the linear component and N_t denotes the nonlinear component. These two components must be estimated from the input data. First, ARIMA models the linear component. Then, using ANNs to model the

residues of the linear model, nonlinear relationships can be discovered.

In summary, the proposed methodology of the ARIMA residual hybrid system consists of two steps. In the first step, an ARIMA model is used to analyze the linear part of the problem. In the second step, a neural network model is developed to model the residuals from the ARIMA model. Since the ARIMA model cannot capture the nonlinear structure of the data, the residuals of the linear model will

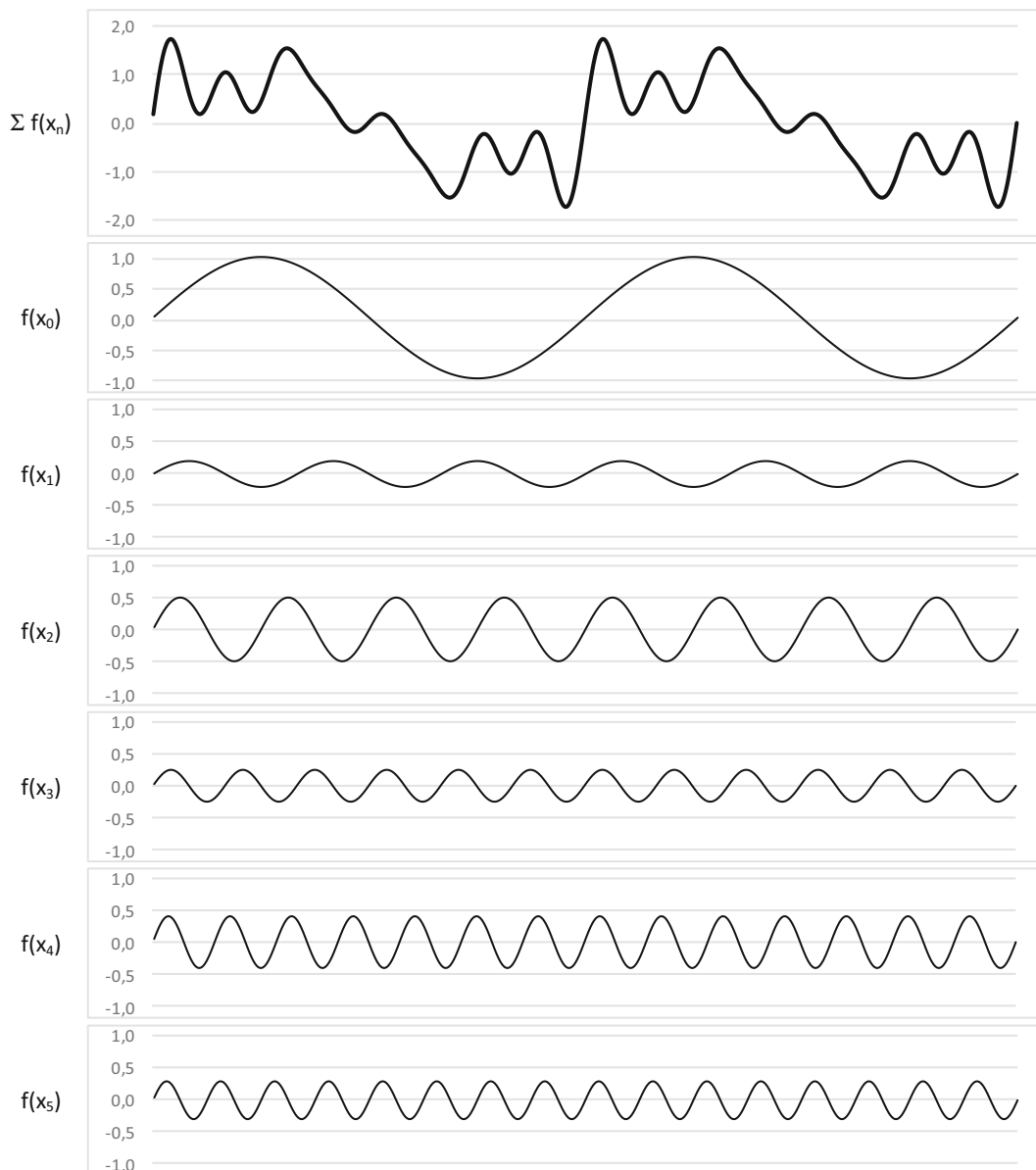


Fig. 12 Discrete wavelet transform

contain information about nonlinearity. The results from the neural network can be used as predictions of the error terms for the ARIMA model. The hybrid model explores the unique feature and strength of the ARIMA model, as well as the ANN model in determining different patterns. Thus, it may be advantageous to model linear and nonlinear patterns separately using different models, and then combine predictions to improve overall modeling and forecasting performance.

5.3.2 ARIMA and ANN with Discrete Wavelet Transform

This model defined by Khandelwal et al. [32] uses discrete wavelet transform (DWT) to obtain the decomposition of a time series into high- and low-frequency components. The waveform shown in Fig. 12 represents any function as a superposition of a set of small waves. Because these functions are small waves, located at different times in the time series, the waveform can provide crucial information about time and frequency domains concerning the source data. The *Haar* and *Daubechies* patterns are commonly used for prediction applications [7, 29] and [45]. To mitigate the problem of selection of the appropriate technique, the proposed model considers the average of the predictions through three waveforms: *Harr*, *db2*, and *db4*.

The proposed approach consists of two phases: decomposition and reconstruction [18]. In the first phase, the series is decomposed into filters that identify the highest (detailed) and lowest (approximate) frequency components of the series, as seen in Fig. 12. In the next phase, the high- and low-frequency components are reconstructed through inverse DWT (IDWT) [7, 18]. After these two phases, an ARIMA is applied in the reconstructed detailed part, and the predictions are generated. Then, an ANN is applied to the corresponding residual along with the approximate part. Finally, combined forecasts are obtained by adding the predictions of the two components.

5.4 Analysis of Load Prediction Methods

The forecasting methods presented in this section are summarized in Table 7. The models are classified as linear, non-linear and hybrid. Each method presents advantages and disadvantages, and has to be applied according to the characteristic of the monitored data. Linear models such MA, AR, ARIMA and Holt-Winters are suitable to deterministic data. The main advantages of this methods are the speed and simplicity in calculation and do not require much data to work. The main disadvantage is the data has to be linear. To model a stochastic non-linear time series, the indicated technique is Artificial Neural Network. The advantage of ANN is the ability to estimate a large

Table 7 Summary of prediction methods

Method	Advantages	Disadvantages
Moving Average (MA)	Simple and fast to model and compute, few data is required	Indicates only the trend of data (slope)
Autoregressive (AR)	Few past data is required to predict the next values	Data has to be stationary and calculation of parameters of the model is not simple
AR Integrated MA (ARIMA)	Considers trend, seasonality, cyclicity and irregularity	Data has to be stationary, not trivial modeling and calculating
Holt-Winters	Suitable for producing a short-term forecast, easy to compute, focus on recent observations, minimum data is required	Not accurate to long-term forecast, not suitable to cyclical or seasonal data variations
Artificial Neural Network (ANN)	Model and predict a stochastic non-linear process, increased precision in forecasting	The network has to be trained, requires a lot of data, model deprecates as the time advances
Hibrid Models	Split the process in stochastic and deterministic components, high precision in forecasting	Deep knowledge of the process behavior, complex statistical processing, requires a lot of data

class of functions with a high degree of accuracy, not requiring prior knowledge of the model. The disadvantages are the great amount of time and data needed to train the network and the depreciation of the model as the input data changes its behaviour. The advantage of hybrid models is the high precision in forecasting, but the disadvantages are the need of deep knowledge about the process behavior, requires a complex statistical processing and consumes a lot of data to create the model.

6 Discussion

This section presents a discussion on four ways that the presented research pillars can be interconnected to provide pertinent solutions from the perspective of system administrators and application developers. The combination of the three topics results in a Global Management View of the monitored computing infrastructure. The four interconnection ways discussed next cover the following points:

- **System Monitoring.** Load imbalance detection used for inference and prediction.
- **Resource Forecasting.** Forecasting loop cycle, including monitoring, load prediction, and resource management.
- **Adaptive Monitoring.** The cycle of learning the system behavior, monitoring, and feedback in case of changes.

- **Resource Granularity.** A discussion about resource granularity and its impact on performance and overhead.

As a transversal feature, we observed that the most used metric is CPU load, which is measured commonly through discrete data collections. In this way, it is usual to use the following equation to reach the capacity available in a given node: $CPU \cdot (1 - load)$. Here, CPU means the theoretical capacity of a node and $load$ refers to the usage of such CPU at a given moment. Assuming the computing infrastructure has a collection of nodes (resources) and a collection of tasks to assign to them, it can use the aforementioned equation for scheduling and load balancing purposes. However, before applying a load balancing technique, the computing infrastructure must decide if the system is balanced or not. It can use (9) and (10) to implement this functionality. Here, *averagetime* means the arithmetic load average of each node, and D is a percentage that indicates how far from the average a process could be and still achieve a balanced stated. Figure 13 illustrates how the computing infrastructure can infer whether a system load is balanced or not. In both extremities, the system is unbalanced, so load balancing must take place to reorganize the load among the computational nodes.

$$time\ of\ the\ slowest\ process < average\ time \cdot (1 + D) \quad (9)$$

$$time\ of\ the\ slowest\ process > average\ time \cdot (1 + D) \quad (10)$$

Fig. 13 Load balance range

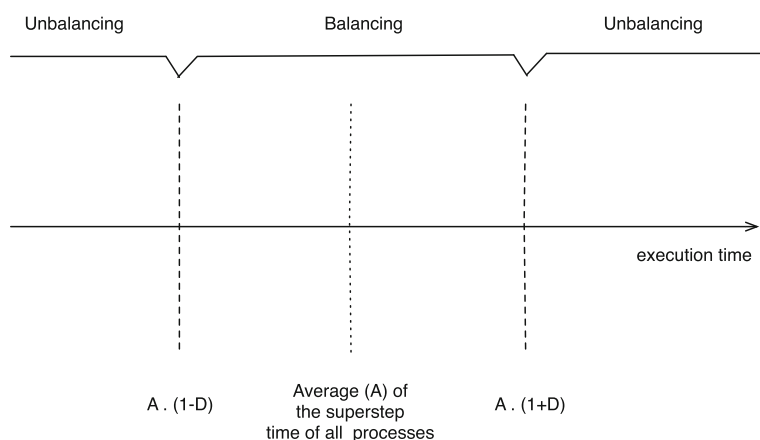
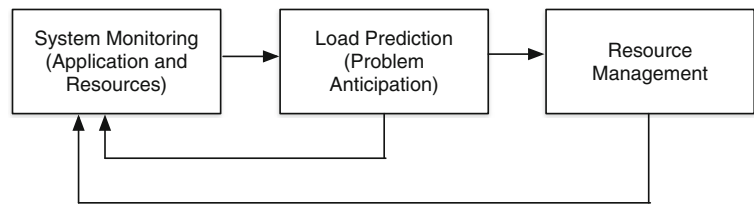


Fig. 14 Closed-loop architecture comprising the three studied research pillars: system monitoring, load prediction and resource management



When observing system monitoring, load prediction, and resource management in a unified perspective, we can model a closed-loop observation architecture. Figure 14 depicts such an architecture and the interaction among the research pillars considered in this survey. The monitoring facility is responsible for taking values, either discretely or continuously, about a particular metric or set of them. The idea is to generate an amount of historical data from which the observation architecture can build knowledge about the system. For example, applying load prediction to know the system behavior in the future given a tracing obtained from the past. Based on forecasting, the load predictor can perceive abnormal situations such as CPU load saturation, disk full, not enough free memory to execute the application, and so on. Third, after detecting and anticipating the occurrence of a problem, the resource manager can employ resource management techniques to either mitigate or nullify such a problem.

Another relationship among the research pillars in this survey considers the interaction presented in Fig. 15. Here, the idea is to train the monitoring system to detect application behavior concerning CPU usage and IO operations. The final output of the first phase is a chart where the x-axis represents the elapsed time while the y-axis refers to the observed metric. The goal of this procedure is to detect periodical application behaviors to feed the next phases. After collecting the application patterns, the system monitors the application and analyze the instant current data against the expected pattern. If a significant discrepancy is detected, notifications take place to the system administrator. Thus, the administrator can act by using one

or more resource management techniques to transform an abnormal situation in a situation that is closer to the predicted pattern. Finally, using a predefined system interval, the system can rebuild the application patterns, enabling the execution of dynamic applications that run different tasks or handle diverse input workloads along the time.

The last point of discussion regards system granularity when using multiple replicas. Granularity affects the performance and overhead of computation division, communication, and monitoring, presenting decision trade-offs. The larger the number of replicas, the better the load balancing. However, as the number of replicas grows, both the communication and content coherence algorithm complexities grow as well. The system and application must decide the most profitable number of replicas given a particular system load. As for monitoring, it is vital to consider the accuracy and freshness of the obtained data, affected by the interval of the data collection. The larger this value, the lesser the system reactivity on identifying an incident and dealing with it using a resource management technique. The shorter the monitoring interval, the larger the monitoring intrusiveness in the executing application performance. Thus, the system faces a trade-off to decide the monitoring interval that will yield the best cost-benefit ratio. In this context, da Rosa Righi et al. [60] propose an adaptive strategy to accomplish this. The algorithm starts with a predefined monitoring interval (e.g., 5 seconds). After 5 seconds of execution, the system is re-analyzed. If there are no incidents (e.g., unbalanced state), this interval doubles to generate less intrusiveness. How-

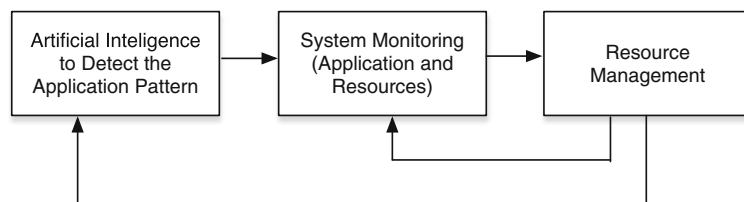


Fig. 15 Detecting the Application Pattern, then performing correction actions by applying system monitoring and resource management. In addition, we reanalyze again in application pattern in accordance with a predefined period

ever, if an incident takes place, the algorithm divides the monitoring interval up to the predefined lower threshold to active mitigation actions faster.

As a combination of system monitoring, resource management and load prediction, here we present three initiatives: AutoElastic [57], MigPF [61] and Proliot [58]. AutoElastic performs periodical monitoring and anomaly detection on the load of high-performance (HPC) applications, so applying Round-Robin-based load balancing with moving average-driven cloud elasticity. MigPF acts a Blackbox middleware, so not needing any modification in the HPC application. MigPF works with process rescheduling on multi-cluster settings, using load balancing with hybrid load prediction techniques to decide which processes will migrate and the future location of them. Finally, Proliot also acts a Black box solution on IoT applications, providing proactive cloud elasticity through machine learning techniques such as ARIMA load prediction. The three aforementioned examples show us how can we efficiently combine different ideas in favor of improving a particular metric. In particular, AutoElastic and MigPF aim at reducing the application time, while Proliot objectives a better IoT transaction throughput.

Regarding limitations of this survey, we discuss the sample of proposals. This survey presents a set of state-of-the-art proposals and studies about each of the three pillars: (i) system monitoring; (ii) resource management; (iii) load prediction. They provide examples of many different techniques, approaches, and methods to address each topic and challenge discussed. However, it is not an exhaustive list, and there may be other proposals that could provide better results for specific scenarios. Without loss of generality, we consider that the selected articles cover a broad area of the topics discussed, providing a diversity of examples. An example of an area which is not effectively is security, which is of paramount importance in any proposed application or system. Regarding enterprise applications, there are many potential security threats regarding private information of users.

7 Future Directions

In this section, we complement the analysis of the state-of-the-art by indicating potential topics to be explored in future work. Such topics were drawn

from the analysis of the surveyed initiatives, specially considering their “Future Work” sections. Thus, the indications are:

- **Monitoring Quality.** Monitoring is a key process to assure the efficient performance of a computing system. A computing infrastructure needs to consider some trade-offs for designing the monitoring workflow, specially in terms of the employed collect method and periodicity. Such trade-offs impact on the consumption of human and computational resources for the monitoring process. For example, the collect method concerns how to gather data from the machines and systems: samples, snapshots, probes, etc. The periodicity refers to the frequency of the monitoring, which can result in a discrete or continuous analysis. Both aspects, collect method and periodicity, have a significant impact in the monitoring quality as a whole.
- **Monitoring Intrusiveness.** Besides specifying parameters for monitoring quality, the system also has to consider its monitoring intrusiveness. The goal is to avoid the disruption of running applications due to the monitoring and processing tasks, maintaining the monitoring coverage and granularity. There are two extremes for such intrusiveness: changing (i.e., annotating) the code for logging and observing the application as a black box. The first option gives better results and granularity for the monitoring metrics at a higher intrusion. The second option provides less insights on the internals of the monitored machine/system, but it can be theoretically applied to any machine/system.
- **Monitoring and Managing Decisions.** This topic is related to the distribution approach used for the monitoring and management tasks: centralized or decentralized. The first approach offers simplicity to implement and higher accuracy. Besides that, centralization can help the use some techniques, such as Machine Learning. However, centralized monitoring brings Single Point-of-Failure (SPoF) and less scalability (i.e., central resources are bottlenecks). On the other hand, the decentralized approach attempts to overcome the scalability shortcomings and to improve the reaction time, specially when applied to large computing systems. In addition, it is possible to integrate the probes in a Multi-Agent System (MAS).

- **Resource Management Automation.** Automation abilities attempt to provide ways for computing infrastructures and applications provisioning to use auto-scaling according to their load policies. For such use, it is necessary to bind system monitoring and load prediction to resource management by automating the resource and application management while maintaining metrics related to the QoS and SLA. The goal is to improve the efficiency of the resource consumption, while avoiding automation decisions which can impact negatively either the computing infrastructure (e.g., waste of resources) or clients (e.g., high cost).
- **Cloud Models.** The use of hybrid clouds is widely spread nowadays, but such clouds are not necessarily the most efficient ones. Private clouds require infrastructure investments and are more costly than public ones. Thus, ideally, enterprises will use the private clouds when handling confidential data only whereas the public cloud, for the rest. However, the sharing of computational load between the private and public models in an efficient way regarding operation and communication costs is a challenging task. Besides that, it is necessary that the load decision to be adaptive in order to cope with highly dynamic environments, such as those found on some commercial systems.
- **Geolocation.** The last mentioned opportunity is about geolocation features and how they could be used for load balancing. Considering that computing infrastructures may be located in different time zones and are affected differently by periodical events, their location could be used to load the balance in a trade-off between processing time and communication time overhead. That is, it could be advantageous sometimes to process data farther than in a closer heavy loaded computing infrastructure. Some efforts are being applied to solve this issues and are available in [36] and [37].

8 Conclusion

This article presented a survey on combining and discussing system monitoring, load prediction, and resource management to provide a global administration viewpoint. The goal was to present different

monitoring strategies and how to orchestrate them to offer better performance for enterprise applications, as well as to mitigate problems through anticipating them using load forecasting. It is important to highlight the possibility of developing and combining these three methods, such as the use of periodic monitoring, Round-Robin load balancing, and the ARIMA prediction can be useful for each situation. In our understanding, future networked-based system must look for system monitoring, resource management, and artificial intelligence, since more and more we have a competitive market where financial costs for system administrators and clients must be taken into account. In addition, the combination of the aforesaid pillars is also useful to improve client loyalty, since we can provide a more customized and QoS-enable service for each of them.

In this way, the contribution of the survey is twofold: (i) to discuss system monitoring at resource and application levels, resource management and load prediction in a single document; (ii) to present the possible relationship among the three research pillars, also highlighting opportunities for future works in the area. The aforementioned contributions can be applied on a series of target applications, such as management of databases and log files, e-commerce, high-performance computing, payment processing, business intelligence (BI) and enterprise resource planning (ERP). We argue that is very important to integrate all counterparts of the application execution, including hardware and software aspects, which are split among monitoring, resource management, and load prediction. In other words, we emphasize that performance and scalability are achieved not only by applying a fine tuning on each of the research pillars, but also on developing interfaces and frameworks to connect them properly.

Acknowledgements This article was partially supported by the following Brazilian agencies: CAPES, CNPq and FAPERGS. In addition, we would like to thank DELL for also supporting this research.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Aaziz, O., Cook, J., Sharifi, H.: Push me pull you: Integrating opposing data transport modes for efficient hpc application monitoring. In: 2015 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, pp. 674–681 (2015)
2. Aceto, G., Botta, A., De Donato, W., Pescapè, A.: Cloud monitoring: a survey. *Comput. Netw.* **57**(9), 2093–2115 (2013)
3. Agarwala, S., Poellabauer, C., Kong, J., Schwan, K., Wolf, M.: System-level resource monitoring in high-performance computing environments. *J. Grid. Comput.* **1**(3), 273–289 (2003)
4. Akbar, M.F., Munir, E.U., Rafique, M.M., Malik, Z., Khan, S.U., Yang, L.T.: List-based task scheduling for cloud computing. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 652–659. <https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.143> (2016)
5. Al-Ayyoub, M., Daraghme, M., Jararweh, Y., Althebyan, Q.: Towards improving resource management in cloud systems using a multi-agent framework. *Int. J. Cloud Comput.* **5**(1–2), 112–133 (2016)
6. Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., Merle, P.: Elasticity in cloud computing: state of the art and research challenges. *IEEE Trans. Serv. Comput.* **PP**(99), 1–1 (2017). <https://doi.org/10.1109/TSC.2017.2711009>
7. Al Wadia, M., Tahir Ismail, M.: Selecting wavelet transforms model in forecasting financial time series data based on arima model. *Appl. Math. Sci.* **5**(7), 315–326 (2011)
8. Alhamazani, K., Ranjan, R., Mitra, K., Rabhi, F., Jayaraman, P.P., Khan, S.U., Guabtni, A., Bhatnagar, V.: An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing* **97**(4), 357–377 (2015)
9. Amiri, M., Mohammad-Khanli, L.: Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications* (2017)
10. Balcas, J., Kcira, D., Mughal, A., Newman, H., Spiropulu, M., Vlimant, J.: Monalisa, an agent-based monitoring and control system for the lhc experiments. In: *Journal of Physics: Conference Series*, IOP Publishing, vol. 898, p. 092055 (2017)
11. Borchert, K., Hirth, M., Zinner, T., Mocanu, D.C.: Correlating qoe and technical parameters of an sap system in an enterprise environment. In: 2016 28th International Teletraffic Congress (ITC 28), IEEE, vol. 3, pp. 34–36 (2016)
12. Bouabdallah, R., Lajmi, S., Ghedira, K.: Use of reactive and proactive elasticity to adjust resources provisioning in the cloud provider. In: 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (2016)
13. Box, G.E., Jenkins, G.M.: Time series analysis forecasting and control. Tech. rep., Wisconsin Univ Madison Dept of Statistics (1970)
14. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. Wiley, New York (2015)
15. Carvallo, P., Cavalli, A.R., Mallouli, W., Rios, E.: Multi-cloud applications security monitoring. In: *International Conference on Green, Pervasive, and Cloud Computing*, Springer, pp. 748–758 (2017)
16. Casavant, T.L., Kuhl, J.G.: A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Trans. Softw. Eng.* **14**(2), 141–154 (1988). <https://doi.org/10.1109/32.4634>
17. Chen, J., Wang, C., Zhou, B.B., Sun, L., Lee, Y.C., Zomaya, A.Y.: Tradeoffs between profit and customer satisfaction for service provisioning in the cloud. In: *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, pp. 229–238. ACM, New York (2011). <https://doi.org/10.1145/1996130.1996161>. <http://doi.acm.org/10.1145/1996130.1996161>
18. Choi, T.M., Yu, Y., Au, K.F.: A hybrid sarima wavelet transform method for sales forecasting. *Decis. Support. Syst.* **51**(1), 130–140 (2011)
19. Duan, R., Prodan, R., Li, X.: Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds. *IEEE Trans. Cloud Comput.* **2**(1), 29–42 (2014). <https://doi.org/10.1109/TCC.2014.2303077>
20. Farshchi, M., Schneider, J.G., Weber, I., Grundy J.: Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software* (2017)
21. Fatema, K., Emeakaroha, V.C., Healy, P.D., Morrison, J.P., Lynn, T.: A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *J. Parallel Distrib. Comput.* **74**(10), 2918–2933 (2014)
22. Fittkau, F., Hasselbring, W.: Elastic application-level monitoring for large software landscapes in the cloud. In: *European conference on service-oriented and cloud computing*, Springer, pp. 80–94 (2015)
23. Frachtenberg, E., Schwiigelshohn, U.: New challenges of parallel job scheduling. In: *Proceedings of the 13th International Conference on Job Scheduling Strategies for Parallel Processing*, vol. JSSPP'07, pp. 1–23. Springer-Verlag, Berlin (2008). <http://dl.acm.org/citation.cfm?id=1791551.1791552>
24. Galante, G., d Bona, L.C.E.: A Survey on Cloud Computing Elasticity. In: 2012 IEEE 5th International Conference on Utility and Cloud Computing, pp. 263–270. <https://doi.org/10.1109/UCC.2012.30> (2012)
25. Galante, G., Erpen De Bona, L.C., Mury, A.R., Schulze, B., Rosa Righi, R.: An analysis of public clouds elasticity in the execution of scientific applications: a survey. *J. Grid Comput.* **14**(2), 193–216 (2016). <https://doi.org/10.1007/s10723-016-9361-3>
26. Ghaderi, J.: Simple high-performance algorithms for scheduling jobs in the cloud. In: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 345–352. <https://doi.org/10.1109/ALLERTON.2015.7447025> (2015)

27. Guan, Q., Zhang, Z., Fu, S.: Proactive failure management by integrated unsupervised and semi-supervised learning for dependable cloud systems. In: 2011 6th International Conference on Availability, Reliability and Security, pp. 83–90. <https://doi.org/10.1109/ARES.2011.20> (2011)
28. Holt, C.C.: Forecasting seasonals and trends by exponentially weighted moving averages. *Int. J. Forecast.* **20**(1), 5–10 (2004). <https://doi.org/10.1016/j.ijforecast.2003.09.015>. <http://www.sciencedirect.com/science/article/pii/S01692070030001134>
29. Hsieh, T.J., Hsiao, H.F., Yeh, W.C.: Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm. *Appl. Soft Comput.* **11**(2), 2510–2525 (2011)
30. Katsaros, G., Subirats, J., Fitó, J.O., Guitart, J., Gilet, P., Espling, D.: A service framework for energy-aware monitoring and vm management in clouds. *Futur. Gener. Comput. Syst.* **29**(8), 2077–2091 (2013)
31. Khan, M., Khendek, F., Toeroe, M.: Monitoring service level workload and adapting highly available applications. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, pp. 522–529 (2016)
32. Khandelwal, I., Adhikari, R., Verma, G.: Time series forecasting using hybrid arima and ann models based on dwt decomposition. *Proc. Comput. Sci.* **48**, 173–179 (2015)
33. Khashei, M., Bijari, M.: A novel hybridization of artificial neural networks and arima models for time series forecasting. *Appl. Soft Comput.* **11**(2), 2664–2675 (2011). <https://doi.org/10.1016/j.asoc.2010.10.015>. <http://www.sciencedirect.com/science/article/pii/S1568494610002759>, the Impact of Soft Computing for the Progress of Artificial Intelligence
34. Krauter, K., Buyya, R., Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience* **32**(2), 135–164 (2002). <https://doi.org/10.1002/spe.432>
35. Liu, J., Pacitti, E., Valduriez, P., Mattoso, M.: A survey of data-intensive scientific workflow management. *J. Grid Comput.* **13**(4), 457–493 (2015)
36. Liu, J., Pacitti, E., Valduriez, P., De Oliveira, D., Mattoso, M.: Multi-objective scheduling of scientific workflows in multisite clouds. *Futur. Gener. Comput. Syst.* **63**, 76–95 (2016)
37. Liu, J., Pacitti, E., Valduriez, P., Mattoso, M.: Scientific workflow scheduling with provenance data in a multisite cloud. In: Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIII, Springer, pp. 80–112 (2017)
38. Liu, J., Pacitti, E., Valduriez, P.: A survey of scheduling frameworks in big data systems. *Int. J. Cloud Comput.* **7**, 1–27 (2018)
39. Ma, H., Wang, L., Tak, B.C., Wang, L., Tang, C.: Auto-tuning Performance of MPI Parallel Programs Using Resource Management in Container-Based Virtual Cloud. In: 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), pp. 545–552. <https://doi.org/10.1109/CLOUD.2016.0078> (2016)
40. Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y., Abdulhamid, S.M.: Resource Scheduling for Infrastructure As a Service (IaaS) in Cloud Computing. *J. Netw. Comput. Appl.* **68**(C), 173–200 (2016). <https://doi.org/10.1016/j.jnca.2016.04.016>
41. Mandal, A., Ruth, P., Baldin, I., Król, D., Juve, G., Mayani, R., Da Silva, R.F., Deelman, E., Meredith, J., Vetter, J., et al.: Toward an end-to-end framework for modeling, monitoring and anomaly detection for scientific workflows. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops. IEEE, pp. 1370–1379 (2016)
42. Manvi, S.S., Shyam, G.K.: Resource management for infrastructure as a service (iaas) in cloud computing: a survey. *J. Netw. Comput. Appl.* **41**, 424–440 (2014)
43. Markham, I.S., Rakes, T.R.: The effect of sample size and variability of data on the comparative performance of artificial neural networks and regression. *Comput. Oper. Res.* **25**(4), 251–263 (1998)
44. Mell, P.M., Grance, T.: SP 800-145. The NIST definition of cloud computing. Tech. Rep. Gaithersburg, United States (2011)
45. Milidui, R.L., Machado, R.J., Renteria, R.P.: Time-series forecasting through wavelets transformation and a mixture of expert models. *Neurocomputing* **28**(1), 145–156 (1999)
46. Morton, A.: Active and passive metrics and methods (with hybrid types in-between). RFC 7799 (Informational) (2016)
47. Netto, M.A.S., Calheiros, R.N., Rodrigues, E.R., Cunha, R.L.F., Buyya, R.: HPC cloud for scientific and business applications: taxonomy, vision, and research challenges. *ACM Comput. Surv.* **1**(1), 1–1 (2017)
48. Pahl, C.: Containerization and the PaaS Cloud. *IEEE Cloud Comput.* **2**(3), 24–31 (2015). 10.1109/MCC.2015.51
49. Patel, D.K., Tripathy, D., Tripathy, C.: Survey of load balancing techniques for grid. *J. Netw. Comput. Appl.* **65**(C), 103–119 (2016). <https://doi.org/10.1016/j.jnca.2016>
50. Pavlou, G.: On the evolution of management approaches, frameworks and protocols: a historical perspective. *J. Netw. Syst. Manag.* **15**(4), 425–445 (2007). <https://doi.org/10.1007/s10922-007-9082-9>
51. Persico, V., Grimaldi, D., Pescapè, A., Salvi, A., Santini, S.: A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Trans. Parallel Distrib. Syst.* **28**(8), 2117–2130 (2017). <https://doi.org/10.1109/TPDS.2017.2651810>
52. di Pietro, A., Huici, F., Costantini, D., Niccolini, S.: Decon: Decentralized coordination for large-scale flow monitoring. In: Proceedings..., Proceedings of the IEEE Conference on Computer Communications (INFOCOM), pp. 1–5. IEEE Computer Society, Washington (2010). <https://doi.org/10.1109/INFCOMW.2010.5466642>
53. Poddar, R., Vishnoi, A., Mann, V.: HAVEN: Holistic load balancing and auto scaling in the cloud. In: 2015 7th International Conference on Communication Systems and Networks (COMSNETS), pp. 1–8. <https://doi.org/10.1109/COMSNETS.2015.7098681> (2015)
54. d R Righi, R., Rodrigues, V.F., da Costa, C.A., Galante, G., de Bona, L.C.E., Ferreto, T.: AutoElastic: Automatic resource elasticity for high performance applications in the cloud. *IEEE Trans. Cloud Comput.* **4**(1), 6–19 (2016). <https://doi.org/10.1109/TCC.2015.2424876>

55. Ranjan, R., Benattallah, B.: Programming cloud resource orchestration framework: operations and research challenges. arXiv:12042204 (2012)
56. Righi, R.D.R.: MigBSP: a new approach for processes rescheduling management on bulk synchronous parallel applications (2009)
57. Righi, R.D.R., Rodrigues, V.F., da Costa, C.A., Galante, G., de Bona, L.C.E., Ferreto, T.: Autoelastic: automatic resource elasticity for high performance applications in the cloud. *IEEE Trans. Cloud Comput.* **4**(1), 6–19 (2016). <https://doi.org/10.1109/TCC.2015.2424876>
58. Rodrigues, V.F., Correa, E., da Costa, C.A., da Rosa Righi, R.: On exploring proactive cloud elasticity for internet of things demands. In: 2017 XLIII Latin American Computer Conference, CLEI 2017, Córdoba, Argentina, September 4–8, 2017, pp. 1–10. <https://doi.org/10.1109/CLEI.2017.8226417> (2017)
59. Röhl, T., Eitzinger, J., Hager, G., Wellein, G.: Likwid monitoring stack: A flexible framework enabling job specific performance monitoring for the masses. In: 2017 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, pp. 781–784 (2017)
60. da Rosa Righi, R., Pilla, L.L., Carissimi, A.S., Navaux, P.O.A., Heiss, H.U.: Applying processes rescheduling over irregular BSP application, pp. 213–223. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-01970-8_22
61. da Rosa Righi, R., de Quadros Gomes, R., Rodrigues, V.F., da Costa, C.A., Alberti, A.M., Pilla, L.L., Navaux, P.O.A.: Migpf: Towards on self-organizing process rescheduling of bulk-synchronous parallel applications. *Futur. Gener. Comput. Syst.* **78**, 272–286 (2018). <https://doi.org/10.1016/j.future.2016.05.004>. <http://www.sciencedirect.com/science/article/pii/S0167739X16301145>
62. Sahi, S.K., Dhaka, V.: A survey paper on workload prediction requirements of cloud computing. In: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, pp. 254–258 (2016)
63. Sawamura, R., Boeres, C., Rebello, V.E.F.: MEC: The Memory Elasticity Controller. In: 2016 IEEE 23rd international conference on high performance computing (HiPC), pp. 111–120. <https://doi.org/10.1109/HiPC.2016.022> (2016)
64. Sekar, V., Reiter, M.K., Willinger, W., Zhang, H., Kompella, R.R., Andersen, D.G.: Csamp: A system for network-wide flow monitoring. In: Proceedings..., USENIX Symposium on Networked Systems Design and Implementation (NSDI), pp. 233–246. USENIX Association, Berkeley (2008)
65. Seneviratne, S., Witharana, S.: A survey on methodologies for runtime prediction on grid environments. In: 2014 7th International Conference on Information and Automation for Sustainability (ICIAFS), IEEE, pp. 1–6 (2014)
66. Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., Zekauskas, M.: A one-way active measurement protocol (owamp). RFC 4656 (Proposed Standard) (2006)
67. Shen, H.: RIAL: Resource intensity aware load balancing in clouds. *IEEE Trans. Cloud Comput.* **PP**(99), 1–1 (2017). <https://doi.org/10.1109/TCC.2017.2737628>
68. Singh, S., Chana, I.: A survey on resource scheduling in cloud computing: Issues and challenges. *J. Grid Comput.* **14**(2), 217–264 (2016)
69. Sun, P., Wu, D., Wei, K., Guo, X.: Bans-based cloud resources monitoring system. In: 2015 8th International Symposium on Computational Intelligence and Design (ISCID), IEEE, vol. 2, pp. 445–448 (2015)
70. Tonouchi, T.: A light-weight application monitoring and statistical debugging for a black-box application. In: 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, pp. 523–526 (2015)
71. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **13**(3), 260–274 (2002). <https://doi.org/10.1109/71.993206>
72. Waraich, S.S.: Classification of Dynamic Load Balancing Strategies in a Network of Workstations. In: 5th International Conference on Information Technology: New Generations (itng 2008), pp. 1263–1265. <https://doi.org/10.1109/ITNG.2008.166> (2008)
73. Watts, J., Taylor, S.: A practical approach to dynamic load balancing. *IEEE Trans. Parallel Distrib. Syst.* **9**(3), 235–248 (1998). <https://doi.org/10.1109/71.674316>
74. Weingärtner, R., Bräschner, G.B., Westphall, C.B.: Cloud resource management: a survey on forecasting and profiling models. *J. Netw. Comput. Appl.* **47**, 99–106 (2015)
75. Winters, P.R.: Forecasting sales by exponentially weighted moving averages. *Manag. Sci.* **6**(3), 324–342 (1960)
76. Xu, X., Chen, Y., Calero, J.M.A.: Distributed decentralized collaborative monitoring architecture for cloud infrastructures. *Clust. Comput.* **20**(3), 2451–2463 (2017)
77. Yagoubi, B., Medebber, M.: A load balancing model for grid environment. In: 2007 22nd International Symposium on Computer and Information Sciences, pp. 1–7. <https://doi.org/10.1109/ISCIS.2007.4456873> (2007)
78. Yoo, W., Sim, A.: Time-series forecast modeling on high-bandwidth network measurements. *J. Grid Comput.* **14**(3), 463–476 (2016)
79. Zhang, G.P.: Time series forecasting using a hybrid arima and neural network model. *Neurocomputing* **50**, 159–175 (2003)
80. Zhang, H., Jiang, G., Yoshihira, K., Chen, H.: Proactive workload management in hybrid cloud computing. *IEEE Trans. Netw. Serv. Manag.* **11**(1), 90–100 (2014). <https://doi.org/10.1109/TNSM.2013.122313.130448>