Automatic Evaluation and Comparison of Pub/Sub Systems Performance Improvements

Víctor Rampérez[1], Javier Soriano[1], David Lizcano[2,*] and Carlos Miguel[1]

[1]*Universidad Politécnica de Madrid (UPM), Madrid, Spain*
[2]*Madrid Open University (UDIMA), Madrid, Spain*
E-mail: v.ramperez@upm.es; jsoriano@fi.upm.es; david.lizcano@udima.es; carlos.miguel.alonso@alumnos.upm.es
[*]*Corresponding Author*

Received 15 October 2021; Accepted 29 January 2022;
Publication 12 April 2022

Abstract

Event-driven architectures are becoming more prevalent recently in multiple technological paradigms, especially in web applications, with message brokers being the cornerstone of these architectures. One of the most relevant implementations of these message brokers are content-based publish/subscribe systems. The performance of these systems is a critical factor for web engineering, since the web applications they support need to be reactive despite increases and fluctuations in workloads. However, an obstacle to the research of these systems is the lack of real and publicly available workloads, due to the privacy issue involved in disclosing the interests (subscriptions) of users and other commercial interests of the companies. In this paper we present a parameterizable automated system designed to syntactically translate workloads from different content-based publish/subscribe systems as a means to increase the availability of public workloads to solve the aforementioned problem. As a case study, we describe the evolution

Journal of Web Engineering, Vol. 21_4, 1055–1080.
doi: 10.13052/jwe1540-9589.2144

of a context-aware content-based publish/subscribe system (i.e. E-SilboPS) designed by the authors, which improves up to 5 times the performance of its previous version by reaching the maximum throughput limited by the physical resources of the hardware where it is deployed, as demonstrated by the conducted quantitative evaluation. Then, we validate the utility of the proposed automated workload generation system by using it to make the performance comparison between this new version E-SilboPS and one of the most cited publish/subscribe systems called PADRES, through a real trace of a massively multiplayer online game (MMOG) generated by the latter.

Keywords: Content-based, publish/subscribe, workload generator, elasticity.

1 Introduction

The publish/subscribe communication paradigm is a type of asynchronous communication that decouples the publisher of the messages from the clients that receive the messages based on the interests they have specified (subscribers), i.e. spatial decoupling. Publish/subscribe systems typically use networks of messaging brokers that serve as middleware between publishers and subscribers. Traditionally, publish/subscribe systems are topic-based, i.e., subscribers' interests are defined on the basis of predefined topics.

Event-driven architectures are becoming more prevalent recently in multiple technological paradigms, especially in web applications engineering, with message brokers being the cornerstone of these architectures [21], as shown in Figure 1. One of the best implementations of these message brokers are content-based publish/subscribe systems (CBPS) because of their ability to allow subscribers to specify their interests and only receive notifications according to those interests, as opposed to the processing overhead that subscribers to topic-based publish/subscribe systems have to perform [15, 19, 20, 23].

Despite their importance, an obstacle to the research of these systems is the lack of real and publicly available workloads, due to the privacy issue involved in disclosing the interests (subscriptions) of users and other commercial interests of the companies [23].

We have been working on a system called E-SilboPS which is a context-aware content-based publish/subscribe middleware specially designed to be elastic due to its scaling algorithm. Its architecture is inspired by other CBPS such as SIENA [6, 8] and E-StreamHub [4]. E-SilboPS has been successfully

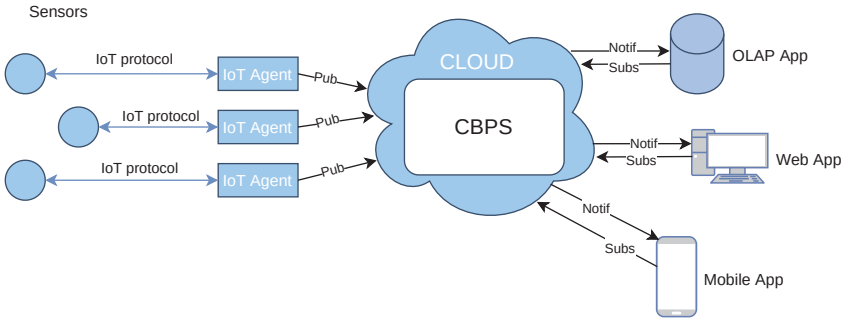

Figure 1 Example of an IoT ecosystem with a content-based publish/subscribe system as a cornerstone for event dispatching. More specifically, end-applications register filters with user interests in the form of subscriptions, and the CBPS system sends corresponding notifications to users when new sensor data (i.e. publications) that match those filters arrive.

used in multiple web application domains, such as Smart Cities and IoT applications [15, 19], and evaluation of auto-scaling systems for multi-cloud environments [16, 17].

This work presents the evolution of E-SilboPS, which improves the performance of the system through changes in its architecture, as well as in the communication between its operator layers, allowing the system to reach the maximum throughput limited by the physical resources of the hardware where it is deployed (up to 5 times faster than its previous version). In addition, to solve the problem of availability of public workloads for content-based publish/subscribe systems mentioned above, a system has been designed and implemented to translate from the format of other content-based workloads to the E-SilboPS syntax automatically.

For the evaluation, several performance tests have been conducted to measure the improvements of the contributions presented in this work. On the one hand, a quantitative evaluation of the performance of the new version of E-SilboPS compared to the previous version has been carried out, showing a throughput increase of up to 5 times compared to the previous version. In addition, to test the workload translator, a real trace of a massively multiplayer online game (MMOG) generated by another content-based publish/subscribe system called PADRES has been used to measure the performance of E-SilboPS with a real workload and compare it with another content-based publish/subscribe system.

The rest of the document is structured as follows: Section 2 presents and analyzes the related work. Section 3 presents the global architecture of E-SilboPS, and Section 4 explains the evolution of the system in the context

of this work. The translator of content-based workloads to the E-SilboPS format is presented in Section 5. The evaluation of this work is explained and analyzed in Section 6, and finally, Section 8 presents the conclusions and future work.

2 Related Work

As already mentioned, publish/subscribe systems implement an asynchronous communication paradigm that allows spatially decoupling the publisher and the subscriber [10]. For this reason, this type of systems have positioned as the cornerstone of event-driven architectures that are gaining so much relevance with technologies such as IoT, Smart Cities or data stream processing [1, 12, 13, 15, 19, 23].

Within the publish/subscribe systems, content-based systems have been gaining considerable relevance in recent years, since they allow improving the limited expressiveness of the topic-based scheme, as well as reducing the overhead of processing subscriptions by subscribers [10].

Some of the most representative publish/subscribe systems that have inspired the system used in this work (E-SilboPS) are SIENA [7], E-StreamHub [4] and PADRES [11].

SIENA (Scalable Internet Event Notification Architectures) [7], is a content-based publish/subscribe system designed to maximize event expressiveness and system scalability. From an architectural point of view, SIENA is composed of a broker network deployed over TCP or UDP. One of the features introduced by SIENA are advertisements. Advertisements are a type of events sent by publishers to declare the type of events they are going to produce. In addition, these events are considered when constructing routing paths, in order to reduce the number of brokers involved in the network. In addition, SIENA offers techniques that constrain the unnecessary propagation of subscriptions by analyzing subscriptions. For all these reasons, the algorithms proposed in SIENA have become the reference for content-based routing system implementations.

PADRES [11], is a content-based publish/subscribe system whose architecture is composed of a peer-to-peer (P2P) overlay network. In addition, PADRES adds a number of additional features to its publish/subscribe system such as: a rule-based matching and routing mechanism, composite subscription processing and historical query-based data access. This content-based publish subscribe system has been used in a wide variety of projects and use cases [5, 25].

E-StreamHub [4] is the elastic version of a scalable, but static content-based publish/subscribe system, called StreamHub [3]. From an architectural point of view, E-StreamHub internally relies on StreamHub, and is composed of 3 layers of operators (access points, matchers and exit points), each of these layers with several slices. According to the authors, the elasticity of E-StreamHub is achieved by the migration of operator slices across a varying number of hosts. However, this does not provide the system with true elasticity since the number of slices of each operator is fixed and defined from the deployment of the system. For the orchestration of all these migration actions, E-StreamHub implements a manager, which is in charge of monitoring the slices and collecting metrics on the resource usage of each host. Finally, an elasticity enforcer is in charge of making scaling decisions, i.e., deciding to which hosts the slices of the most saturated hosts should be migrated in order to improve performance based on predefined policies.

Despite their importance, an obstacle to the research of these systems is the lack of real and publicly available workload traces, due to the privacy issue involved in disclosing the interests (subscriptions) of users and other commercial interests of the companies. The authors of [23] note this problem and address it by proposing a wide-area workload generator for content-based publish/subscribe systems. For this purpose, both subscriber interests and geographic locations are generated through statistical summaries of public data traces. However, despite indicating its intention to make this generator public, it is not currently available. From the best of our knowledge, this is the only proposal for a workload generator based on real traces.

3 E-SilboPS: Architecture Overview

This section presents an overview of the E-SilboPS system architecture.

As already mentioned, E-SilboPS is a context-aware content-based publish/subscribe middleware specially designed to be elastic. This middleware is the elastic version of SilboPS [19]. In addition, it has acquired great relevance as shown by its integration with technological paradigms that demand high performance and are transforming our lives such as Internet of Things (IoT), Smart Cities, Fog Computing [15–17, 20], and more generally, in any event-driven architecture [21].

Inspired by StreamHub's architecture [2], E-SilboPS is composed of four operator layers. Each of these layers is formed by a variable number of instances of the corresponding operator. Figure 2 shows this architecture as well as its operational flow, which is detailed below.

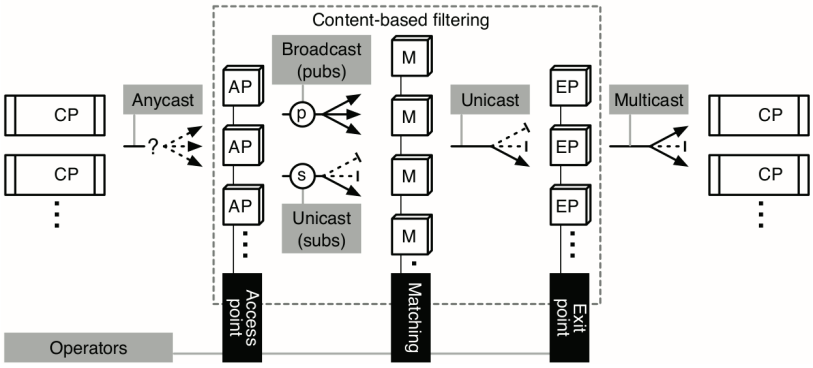

Figure 2 Extracted from [2]. StreamHub and E-SilboPS architecture and operational flow.

Connection Points (CP) are the entry and exit points of the system, since they are in charge of managing the connections with the clients and sending the notifications to the corresponding interested subscribers.

- Subscriptions are forwarded to any *Access Point* (anycast).
- Notifications are forwarded to any *Access Point* (anycast).

Access Points (AP) forward messages, whether notifications or subscriptions, to the *Matchers* layer. More specifically:

- Subscriptions are forwarded to a specific *Matcher* instance (unicast) determined by a selector.
- Notifications are forwarded to all the *Matchers* (broadcast).

Matchers (M) store incoming subscriptions and process the notifications, that is:

- Subscriptions are stored in the internal structure.
- Notifications are matched against the stored subscriptions and dispatched the local result set to the corresponding *Exit Point* (unicast).

Exit Points (EP) forward the notifications to the *Connection Points*.

- Subscriptions never reach this point.
- Collects all partial result sets from the matchers to merge them into a final result set composed of all *Connection Points* instances to which notification must be sent.

Each of these operator layers can scale independently by adding and/or removing instances of those operators to match the demand for the workload at each moment. In this way, the system has a global property of elasticity [19], since it can start from a basic topology and scale out/in each one of the operator layers to be able to optimally manage the workload at each moment, avoiding the need to size the system *a priori* from the beginning.

One of the most valuable features of this system is, undoubtedly, its scaling algorithm that "is able to provide an uninterrupted service to end users even during the scaling operation since its internal state repartitioning is transparent for publishers or subscribers". Moreover, "the scaling operation is time-bounded and depends only on the dimension of the state partitions to be transmitted to the different nodes". More specifically, this algorithm is based on the Chandy-Lamport algorithm [9], commonly known as the snapshot algorithm, which allows to create a consistent cut of the overall system state. Figure 3 summarizes the message exchange during the scale-out operation of the *Matcher* layer. All information regarding the scale-out and scale-in algorithms of each of the operator layers can be found in [19]. As shown in Figure 3 by dashed lines, some coordination between the different layers is necessary to maintain consistency of system state. In particular, the implementation uses Zookeper[1] to ensure distributed coordination between the operator slices.

Throughout this document the form *X-Y-Z* is frequently used to describe the E-SilboPS topology with *X*, *Y* and *Z* being the number of instances of corresponding *AP/CP*, *Matcher* and *EP*, respectively.

4 Improving E-SilboPS Performance

This paper focuses on E-SilboPS, a CBPS system specifically designed to be highly scalable. E-SilboPSS is one of the most recent, high-impact CBPS systems that is attracting a lot of attention recently, as demonstrated by its use in numerous real-world IoT, Smart Cities and Fog Computing contexts [15, 20]. Moreover, high-impact work has recently been carried out to provide this system with real elasticity through the use of auto-scaling systems that combine predictive and reactive techniques [16,17]. For all these reasons, this CBPS system is a clear exponent of the current state of development of this type of system, and is therefore of interest for further study.

[1] https://zookeeper.apache.org/

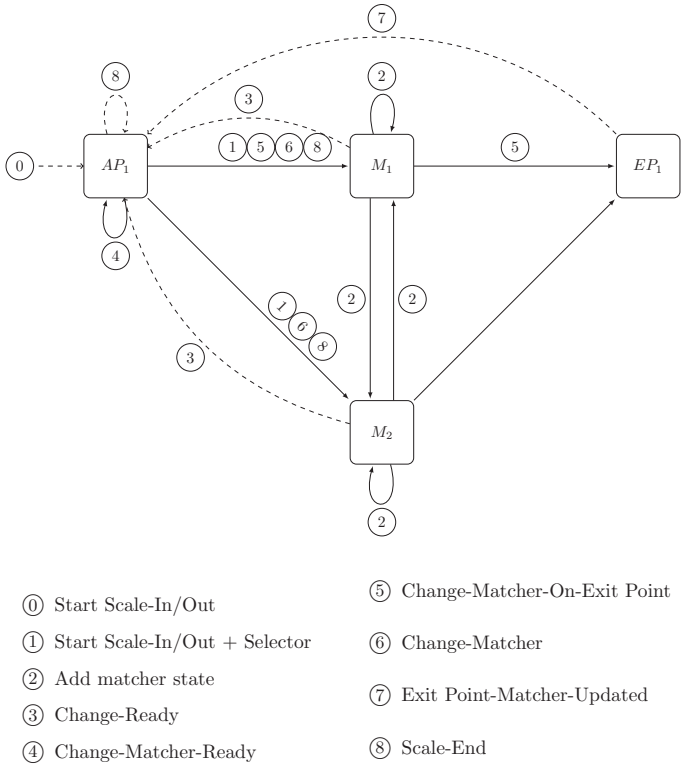

Figure 3 Exchange of messages during the scale-out of the Matchers layer (being $M2$ the new *Matcher* instance deployed). Solid lines represent the communications established between operators for sending notifications and subscriptions. The dashed lines represent the messages used by the Distributed Coordinator to synchronize the layers.

Furthermore, in this work, a case study has been chosen, since this method proposes an explanatory approach to events that cannot be manipulated (i.e. real workload), as described by Yin in [22]: "a case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident".

Some important updates have been made in the context of this work to improve the performance of E-SilboPS with respect to the latest publications of this system [19]. Therefore, this section is dedicated to explain these changes and how they impact on the performance of the system. As explained, E-SilboPS is one of the systems on which the proposals of this

work are evaluated, mainly at the performance level. This section is intended to illustrate these changes and to provide the reader with a baseline of system performance. The quantitative evaluation carried out to assess the enhancement of system performance associated with the improvements presented in this section is described in detail in Section 6.1.

In any system, the maximum throughput is limited by the lowest throughput of each of its parts. In other words, the maximum throughput of the system is that of the component that acts as the bottleneck. In the case of E-SilboPS, the maximum throughput is marked by the component responsible for the highest processing load, which in this case corresponds to the processing of the notifications to determine which subscriptions are matched. In other words, the theoretical bottleneck of the system was in the *Matcher* instances, more specifically, in the processing of notifications, and therefore the maximum throughput of the system would be the maximum throughput of these components.

Note that the concrete values are meaningless as they depend on the infrastructure where the system is running, but let's take the concrete values obtained from the experimentation in the circumstances described in [19] to be able to compare and better understand the performance differences. At the beginning of this work, and using the previous test environment, it was empirically measured that the maximum throughput of the notification processing of a matcher instance was 1,2M notifications/s, and therefore it could be expected that the maximum throughput of the whole system was also 1,2M notifications/s. However, it was empirically proven that the system, even with the simplest 1-1-1 topology was far from reaching that global throughput (i.e. 20k notifications/s on average at most). The rest of the section is devoted to explaining the reasons for this behavior and how they were solved.

On the one hand, to try to mitigate this loss of performance on the one hand, some aspects of implementation (at the code level) were improved, and the serialization of messages was identified as the cause of this loss of throughput. To solve this last issue, other serialization libraries were tested such as *Kryo*[2] and *FAST*[3] and the performance they offered was analyzed. Based on the results obtained from this analysis, it was decided to use the FAST library for the serialization of the messages that were sent by the connections between the instances of the operators.

[2]https://github.com/EsotericSoftware/kryo

[3]https://github.com/RuedigerMoeller/fast-serialization

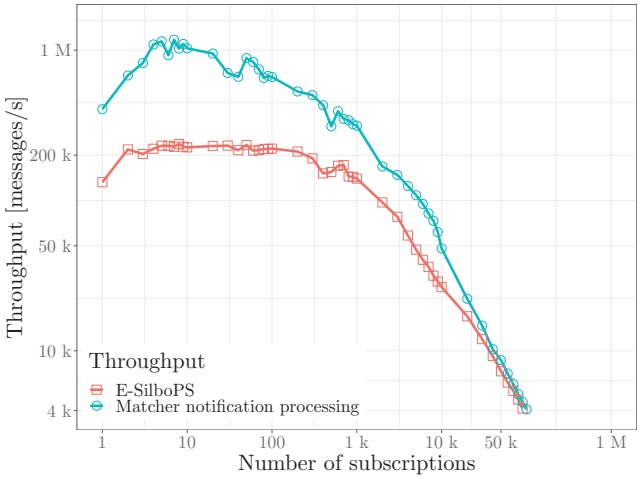

Figure 4 Comparison of E-SilboPS throughput and notification processing Note that the E-SilboPS throughput is that of a single connection, regardless of the topology used.

However, this remarkable improvement only increases the throughput of the connections, but it is not enough to make the global throughput of the system equal to the maximum throughput of the *Matcher's* notifications processing.

This can be clearly seen in Figure 4, which shows how the maximum throughput of *Matcher*'s notifications processing reaches a maximum of 1,2M messages/s, while the global throughput of E-SilboPS is the maximum of a single connection, approximately a maximum of 200k message/s. Of course, this would not be a problem in a 1-1-1 topology, since each operator layer has only one connection, and therefore that connection is the bottleneck, explaining that the total throughput of the system would be that of that connection. However, it was experimentally proven that this same behavior occurred with other topologies such as 3-1-3, in which the expected behavior would be that the global throughput would ideally be multiplied by 3, since it has 3 parallel connections between the *AP/CP-Matcher* and *Matcher-EP* layers.

By combining performance testing and architecture analysis, the problem was isolated in the *Matcher* architecture, more specifically in the connections to the *Exit Points* layer. More specifically, the problem was that by having a single output buffer, it was not possible to parallelize the sending of the local result sets of subscriptions that matched the incoming notification to the different *Exit Points* instances. Finally, the solution adopted is presented in

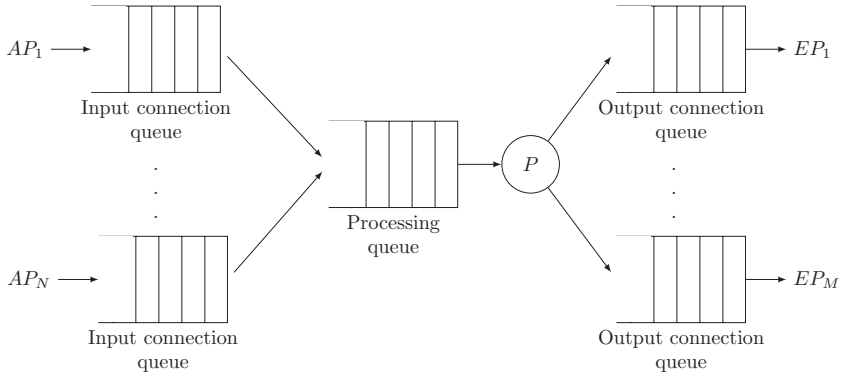

Figure 5 Improved architecture of the E-SilboPS *Matcher* with an output buffer for each *Exit Point* connection.

Figure 5, and consists in adding an output buffer for each connection to an instance of the Exit Points, so that it is possible to parallelize the sending in the same way as in the connections between the layers of *CP/AP-Matcher* operators.

Obviously, in order to achieve the theoretical maximum throughput, a suitable E-SilboPS configuration is necessary. The following rule serves as a first approximation for the calculation of such a configuration:

let $Matcher_{max\ th}$ be the maximum throughput for processing notifications of a *Matcher* instance, i.e. the number of notifications (set P of publications that match with any subscription of the subscription set S) that the matcher is able to calculate per unit of time:

$$Matcher_{max\ th} = \frac{|Notifications|}{\Delta t}$$

$$Notifications = \{P \mid match(P, S)\}$$

and $E\text{-}SilboP_{max\ th}$ be the maximum throughput of the system (i.e. the theoretical maximum throughput of the system), calculated as

$$E\text{-}SilboPS_{max\ th} = X \times Matcher_{max\ th}$$

and $connection_{th}$ the maximum throughput of a connection, i.e. the maximum number of notifications it is capable of sending per unit of time (which is often limited by factors such as notification serialisation);

$$connection_{th} = \frac{notifications_{sent}}{\Delta t}$$

then for the system to reach $E\text{-}SilboPS_{max\ th}$ an $N\text{-}X\text{-}N$ configuration is needed, being X the number of matchers deployed and N calculated as

$$N = \frac{E\text{-}SilboPS_{max\ th}}{connection_{th}}$$

With the data offered above and let

$$\begin{aligned} X &= 1 \\ Matcher_{max\ th} &= 1,2M\ messages/s \\ E\text{-}SilboP_{max\ th} &= Matcher_{max\ th} = 1,2M\ messages/s \\ connection_{th} &= 200k\ messages/s \end{aligned}$$

then

$$N = \frac{1,2M\ messages/s}{200k\ messages/s} = 6$$

Therefore, the minimum deployment configuration of E-SilboPS, in this case, should be 6-1-6.

The quantitative evaluation of all the improvements explained in this section can be found in Section 6.1.

5 Adapting Real Content-based Publish/Subscribe Workloads

As mentioned above, one of the main obstacles to research for content-based publish/subscribe systems is the scarcity of datasets that allow reproducing real workloads, due to privacy issues and commercial interests (especially in subscriptions). Therefore, in this section we present a parser that allows to translate real workload traces from other content-based publish/subscribe systems into E-SilboPS syntax.

As shown in Figure 6, the system receives as input two files: the workload in the original format and a configuration file. Among the configuration options, the most relevant are:

- Mapping between the operators supported by the E-SilboPS syntax and the format of the workload input. Not all E-SilboPS operators have to be supported, only those that appear in the trace and have their correspondence with those of E-SilboPS. Table 1 shows the set of operators supported by the E-SilboPS syntax.

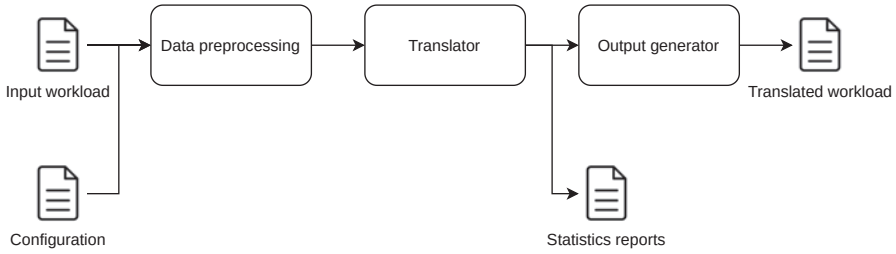

Figure 6 Overview of the content-based workload translator architecture.

Table 1 Set of operators supported by E-SilboPS messages

E-SilboPS	Description
{"exists": ...}	The attribute exists
{"=": ...}	Equal
{"!=": ...}	Not equal
{">": ...}	Greater than
{">=": ...}	Greater or equal
{"<": ...}	Less than
{"<=": ...}	Less or equal
{"^": ...}	Starts with a prefix
{"$": ...}	Ends with a suffix
{"contains": ...}	Contains a string

- Regular expression of publications and subscriptions according to the syntax of the workload input.
- Indicator of whether the trace contains timestamp or any field that allows recreating the order of the workload events, since the translator allows reordering events. If this option is not provided, the reading order of the workload input events is assumed.
- Other options such as: (i) if the input workload includes unsubscriptions and how they should be treated; (ii) if the input workload contains advertisement type events (if so, they are removed in the pre-processing since E-SilboPS does not support these events). Other pre-processing options.

The first module of this translator is responsible for preprocessing the input workload according to the configuration provided. At this stage, advertisement type events (if any) or any other event that does not comply with the format indicated in the configuration are suppressed. Also at this stage the events are reordered based on the specified field (if applicable).

```
s [class,eq,broadcast] (rmi://10.0.1.1:1099/Broker1-M23)
```

(a)

```
{
        "eventType" : "S",
        "id" : "rmi://10.0.1.1:1099/Broker1-M23",
        "contextFunction" : {},
        "constraints" : {
        "class:str" : [{"=" : "broadcast"}]
        }
}
```

(b)

Figure 7 (a) Example of subscription in content-based publish/subscribe PADRES system format. (b) The same subscription obtained as output from the translator for E-SilboPS.

The next module then takes the intermediate file produced by the previous module and translates it into the E-SilboPS format. In addition, this module outputs a statistics file with data on the number of subscriptions, publications, repeated events, etc.

Finally the output generator returns a file in the specified format (e.g. json) with the input workload translated into the E-SilboPS format. This file is ready to be consumed directly by producers and subscribers connected to any E-SilboPS instance. Figure 7 shows an example of translation, with Figure 7(a) being a subscription type event in the PADRES format (input) and its corresponding translation in the E-SilboPS format as output from the translator is shown in Figure 7(b).

Although in this particular use case it has been used to translate workloads from PADRES syntax to E-SilboPS, through the parameterizable configuration file, it is possible to adapt the system to allow translating workloads between different publish/subscribe systems, as long as they are compatible. This fills the gap of being able to test these systems with real workloads, and is a first step to establish benchmarks to compare the performance of this type of systems.

6 Evaluation

This section presents the evaluation conducted, as well as the results and their analysis. More specifically, the first part focuses on the evaluation of

the performance improvement of the new version of E-SilboPS explained above, compared to the previous version of E-SilboPS. On the other hand, the second part of the evaluation makes use of the previously presented content-based workload translator to adapt a real workload used by the PADRES system in the context of a massively multiplayer online games (MMOGs) to the E-SilboPS format and compare the performance of both publish/subscribe systems.

All experiments have been run on an Intel Core i7-4790K 4.00GHz and 16 GB of RAM running Linux 5.3.0-28-generic and openjdk 11.0.7.

6.1 Performance Analysis of the New E-SilboPS Version

As it is a distributed system, as explained above, the first point that could explain this loss of performance was the connections between the different operators' instances. The performance tests demonstrated this hypothesis, as the maximum throughput of a connection was the same as that of the entire system (i.e. 20k notifications/s on average at most).

Since the objective of this evaluation is to compare in terms of performance the previous and the improved versions of E-SilboPS, the same workload used in the performance evaluation of the previous version [19] has been used. More specifically, a synthetic workload composed of publications and notifications has been used, ensuring that the latter always matche with at least one subscription (this is a worst-case scenario, since in reality this is not always true). This workload, composed of 1 million subscriptions and 100k notifications, is created before running the experiment to ensure that the creation of the workload does not impact system performance.

Figure 8 shows the performance (throughput) of the new version of E-SilboPS (code level improvements and the use of the FAST library for message serialization) with different deployment configurations. On the other hand, Figure 9 shows a comparison of the throughput between the new and the old version of E-SilboPS using different deployment configurations. As can be seen, the new E-SilboPS version of this work achieves a throughput improvement of up to 5 times.

However, this remarkable improvement only increases the throughput of the connections, but as it can be seen, it is not enough to make the global throughput of the system equal to the maximum throughput of the *Matcher's* notifications processing.

By means of the improved architecture of E-SilboPS presented in the last part of Section 4, the global throughput of the system was able to reach

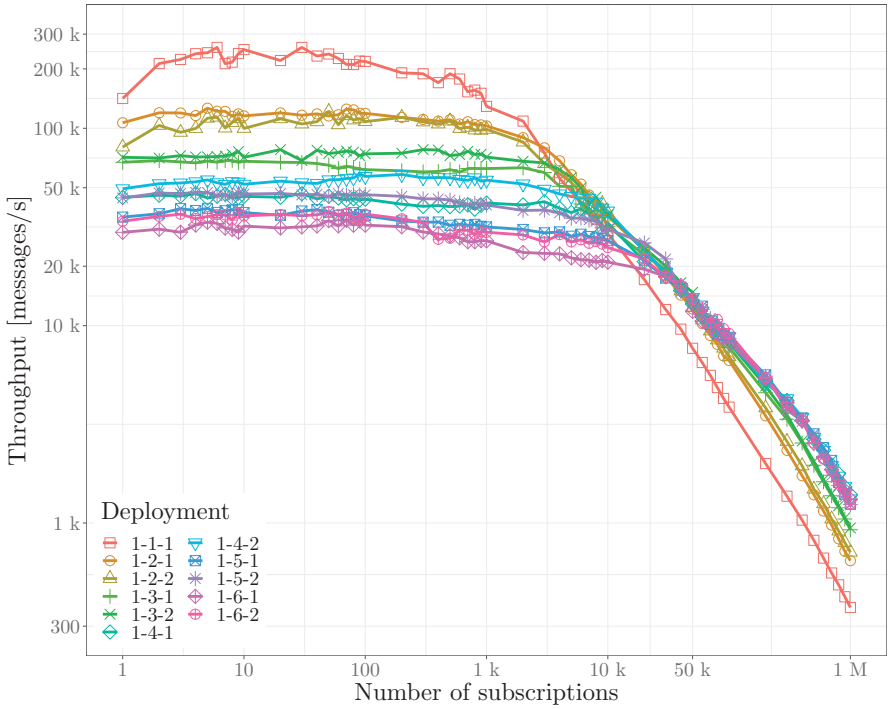

Figure 8 Throughput of the new E-SilboPS version with different deployment configurations.

the maximum throughput of *Matcher*'s notification processing, which is the upper limit that the system can theoretically give. In other words, by improving the performance of serialization and the architecture of the Matcher, it has been possible, with the appropriate deployment configuration, to overcome the bottleneck of the performance of the connections to achieve the maximum throughput, limited by the processing capacity (i.e. CPU bound) and not by the performance of the connections of the distributed system. This allows E-SilboPS to offer maximum performance that only depends on the infrastructure on which it is deployed, not being limited internally at the system level by other bottlenecks.

It is important that the E-SilboPS configuration is the minimum necessary at any given time to avoid wasting resources, since a higher than necessary number of N instances of *CP/AP* and *EP* is an unnecessary extra cost, while the same maximum performance could be obtained with an N-X-N configuration without wasting resources. On the other hand, it is necessary

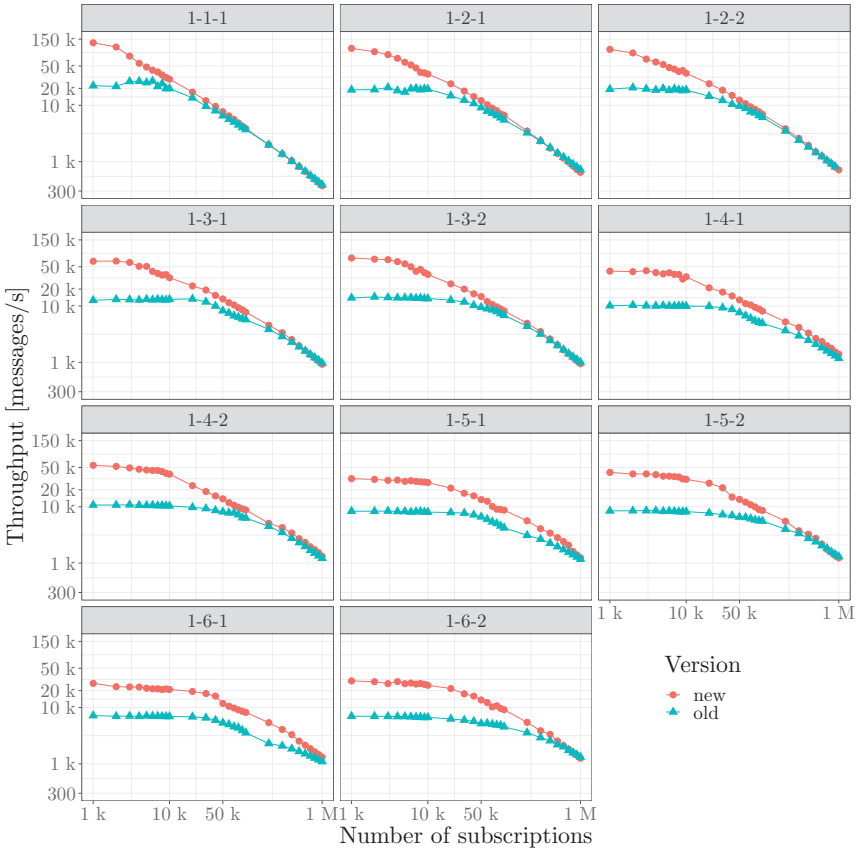

Figure 9 Comparison of new and old E-SilboPS versions throughput with different deployment configurations.

to ensure that resources are sufficient, since an ideal N-X-N topology will not be able to deliver maximum performance if one of the resources is scarce (e.g. CPU saturated at 100% utilization). This may seem obvious, but it is of vital importance in Cloud environments.

All this developed work has allowed to have an extensive knowledge of E-SilboPS and its performance. This knowledge about the implementation and performance of the solution, together with the fact that it offers a similar performance to the existing commercial solutions at the time of writing of this work, have been the reasons for choosing E-SilboPS as an archetype of a distributed system typically deployed in a cloud environment for the evaluation of part of this work.

6.2 Performance Evaluation of the New Version of E-SilboPS with Real Workloads

In this experiment, the real trace of a PADRES content-based workload has been used to evaluate the correct operation of the workload translator presented above. The workload of this experiment is a trace of the content-based publish/subscribe PADRES system obtained from the execution of this system in the context of a research framework, more specifically, a massively multiplayer online games (MMOGs) called MAMMOTH[4] [24].

In the preprocessing phase of the translator, advertisement type events have been removed, the trace has been sorted by the timestamp it provided to recreate the original order of events, error messages have been removed from the trace and a plugin has been developed for mapping subscriptions and unsubscriptions. Subsequently, the output of the translator has been loaded into E-SilboPS to evaluate its performance in terms of performance and to compare it with its PADRES counterpart.

As output of the translator, a single file has been generated in the E-SilboPS json format with all the events sorted to recreate the original trace of the events. Additionally, the most relevant information obtained from the statistics file returned by the system is the following:

- Number of publications: 6.207.207
- Number of subscriptions: 113.904
- Number of unsubscriptions: 112.682
- Duration of the trace: 2 hours and 20 minutes

The results of the execution of this workload are shown in Figure 10. More specifically, the system performance (response time and throughput) is shown with different event input rates (measured in events/s). As can be seen, with low input rates (i.e. 100k and 200k events/s) the system is able to offer excellent performance (very low response time and throughput close to the input rate[5]). However, with very high input rates, simulating work peaks, the system performance degrades (the response time grows and the throughput remains stable at the saturation throughput value around 150k notifications/s).

[4]The original trace used in this experiment can be consulted at http://msrg.org/datasets/mammoth

[5]It may seem surprising that when the system is not saturated, the throughput is not equal to the input rate. However, this is because the input rate is measured in events/s, which includes publications and subscriptions, and the throughput is measured in notifications/s. Therefore, this difference is due to the fact that not all events generate a notification (i.e. publications that do not match any subscription and subscriptions).

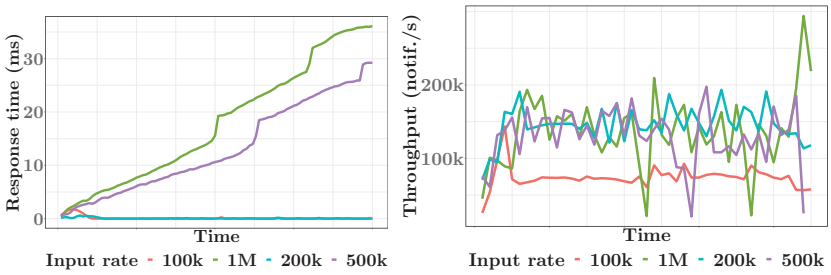

Figure 10 Performance results of the execution of content-based workload of PADRES in E-SilboPS using the workload translator system.

However, even in the worst cases (i.e. input rates of 500k and 1M events/s) it should be noted that although performance is degraded, the values are quite high and sufficient in most application contexts (response time of 35 ms and throughput of 150k notifications/s). Moreover, this experiment has been executed with the minimum configuration of E-SilboPS, i.e. a 1-1-1 topology; performance can be improved by scaling out its operator layers, as demonstrated in the evaluation of the previous section.

7 Discussion

This section will discuss the main implications of the presented work, as well as possible ethical implications and considerations to be taken into account when interpreting the presented evaluation data.

This work is a first step towards a comparative or benchmark study of the different content-based publish/subscribe systems currently in existence. More specifically, the translator presented in this work, which allows workloads to be translated between the different CBPS syntaxes, is an essential step towards establishing benchmarks in this type of system, as it allows the same workload to be adapted to the syntax of different CBPSs. This type of benchmarking was not previously possible, and was a burden when comparing the performance of different CBPS systems.

In this sense, the results shown in this work are a first approximation to demonstrate (i) the validity of the translator presented and (ii) to demonstrate the possibility of comparing the performance of two CBPS systems. However, further effort is required from the developers of these systems to establish a set of standardised workloops for benchmarking.

Finally, the performance improvement of the CBPS system under study in this work (E-SilboPS) may have implications for the design of future similar

systems, since the architecture of this system (based on layers of operators each with a variable number of instances) is being widely used nowadays as a consequence of the importance of the scalability of current cloud systems and the rise of auto-scaling systems (e.g. Kubernetes).

8 Conclusions and Future Work

This paper has presented and described the evolution of E-SilboPS, a context-aware content-based publish/subscribe system. This evolution consists of an improvement of the architecture and the serialization used to maximize the bandwidth of the connections between the different operators that compose the topology of this distributed system. In this way, it has been achieved that the total throughput of the system is the maximum possible with the appropriate topology, i.e., only limited by the hardware resources of the machines where it is deployed. In addition, this improvement has been empirically evaluated through performance tests with different topologies that have shown how the new version of E-SilboPS significantly improves its predecessor (up to 5 times more throughput).

In addition, to fill the gap of the lack of real content-based workloads (due to privacy issues and commercial interests), this work has also presented a content-based workload generator that allows to translate traces between the different syntaxes of content-based publish/subscribe systems. To test the correct functioning of this system, a real trace of a massively multiplayer online game (MMOG) generated by a content-based publish/subscribe system (i.e. PADRES) was used and transformed into the E-SilboPS format. This has allowed to evaluate the performance of E-SilboPS compared to PADRES when executing the same workload, which is a considerable advance for the benchmarking of this type of systems.

On the other hand, a feature that would provide E-SilboPS, and distributed systems in general, with elasticity in cloud environments would be an auto-scaling system. In [17], we have presented a first proposal, but it would be of great value to analyze other predictive methods that allow predicting the behavior of different performance parameters (Service Level Agreement parameters or Key Performance Indicators) and extract conclusions on which predictive methods are more accurate for which metrics, following a similar approach to the one presented in [14].

Furthermore, we are working on additional configuration templates to PADRES for the workload translator. In this way, we intend to offer a set of

configuration templates covering the main content-based publish/subscribe systems, allowing to transform workloads between any pair of these systems.

Finally, a feature that would add great value would be to provide E-SilboPS with an event dispatch traceability system in certain contexts. For this, decentralized technologies such as distributed ledgers (i.e. blockchain) could be used. In this way, in the required context, a record of the events transmitted by E-SilboPS could be stored, making it possible to trace which notifications were sent to which subscriber as a consequence of a publication.

References

[1] Aleksandar Antonić, Martina Marjanović, Krešimir Pripužić, and Ivana Podnar Žarko. A mobile crowd sensing ecosystem enabled by cupus: Cloud-based publish/subscribe middleware for the internet of things. *Future Generation Computer Systems*, 56:607–622, 2016.

[2] Raphaël Barazzutti, Pascal Felber, Christof Fetzer, Emanuel Onica, Jean-françois Pineau, Marcelo Pasin, Etienne Rivière, and Stefan Weigert. StreamHub. In *Proceedings of the 7th ACM international conference on Distributed event-based systems – DEBS '13*, page 63, New York, New York, USA, 2013. ACM Press.

[3] Raphaël Barazzutti, Pascal Felber, Christof Fetzer, Emanuel Onica, Jean-François Pineau, Marcelo Pasin, Etienne Rivière, and Stefan Weigert. Streamhub: A massively parallel architecture for high-performance content-based publish/subscribe. pages 63–74, 06 2013.

[4] Raphaël Barazzutti, Thomas Heinze, Andre Martin, Emanuel Onica, Pascal Felber, Christof Fetzer, Zbigniew Jerzak, Marcelo Pasin, and Etienne Riviere. Elastic Scaling of a High-Throughput Content-Based Publish/Subscribe Engine. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 567–576. IEEE, jun 2014.

[5] Cesar Canas, Kaiwen Zhang, Bettina Kemme, Joerg Kienzle, and Hans-Arno Jacobsen. Publish/Subscribe Network Designs for Multiplayer Games. In *ACM/IFIP/USENIX 15th International Conference on Middleware*, 2014.

[6] Antonio Carzaniga, David S Rosenblum, and Alexander L Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, aug 2001.

[7] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, August 2001.

[8] Antonio Carzaniga and Alexander L. Wolf. Forwarding in a content-based network. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications – SIGCOMM '03*, page 163, New York, New York, USA, 2003. ACM Press.

[9] K. Mani Chandy and Leslie Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, 1985.

[10] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, June 2003.

[11] E. Fidler, Hans-arno Jacobsen, Guoli Li, and Serge Mankovskii. The padres distributed publish/subscribe system. pages 12–30, 01 2005.

[12] Minhyeop Kim, Jung-Hyun Kwon, Hyeon-Jun Jo, and In-Young Ko. Lightweight messaging for efficient service discovery in mobile iot environments using hierarchical bloom filters. *Journal of Web Engineering*, 03 2020.

[13] Bin Lin. Research on data release and location monitoring technology of sensor network based on internet of things. *Journal of Web Engineering*, pages 689–712, 2021.

[14] Ali Yadavar Nikravesh, Samuel A. Ajila, and Chung-Horng Lung. Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 35–45, 2015.

[15] Víctor Rampérez, Javier Soriano, and David Lizcano. A Multidomain Standards-Based Fog Computing Architecture for Smart Cities. *Wireless Communications and Mobile Computing*, 2018:1–14, sep 2018.

[16] Víctor Rampérez, Javier Soriano, David Lizcano, Shadi Aljawarneh, and Juan A. Lara. From sla to vendor-neutral metrics: An intelligent knowledge-based approach for multi-cloud sla-based broker. *International Journal of Intelligent Systems*, n/a(n/a).

[17] Víctor Rampérez, Javier Soriano, David Lizcano, and Juan Lara. Flas: A combination of proactive and reactive auto-scaling architecture for distributed services. *Future Generation Computer Systems*, 118, 01 2021.

[18] Sergio Vavassori. *A Novel Approach to Context-Aware Content-Based Middleware*. PhD thesis, Universidad Politécnica de Madrid, 2016.

[19] Sergio Vavassori, Javier Soriano, and Rafael Fernández. Enabling large-scale iot-based services through elastic publish/subscribe. *Sensors*, 17(9), 2017.
[20] Sergio Vavassori, Javier Soriano, David Lizcano, and Miguel Jiménez. Explicit Context Matching in Content-Based Publish/Subscribe Systems. *Sensors*, 13(3):2945–2966, mar 2013.
[21] W. Roy Schulte Yefim Natis, Massimo Pezzini, Keith Guttridge. The 5 Steps Toward Pervasive Event-Driven Architecture. Technical Report June, Gartner, 2019.
[22] R.K. Yin and SAGE. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications, 2003.
[23] Albert Yu, PK Agarwal, and Jun Yang. Generating wide-area content-based publish/subscribe workloads. *Network Meets Database (NetDB)*, 2009.
[24] Kaiwen Zhang, Cesar Canas, Bettina Kemme, Jörg Kienzle, and Hans-arno Jacobsen. Publish/subscribe network designs for multiplayer games. 12 2014.
[25] Kaiwen Zhang, Mohammad Sadoghi, Vinod Muthusamy, and Hans-Arno Jacobsen. Efficient Covering for Top-k Filtering in Content-Based Publish/Subscribe Systems. In *Middleware*, 2017.

Biographies

Víctor Rampérez is Assistant Professor of Computer Science at Universidad Politécnica de Madrid, UPM, Spain. He holds a B.Sc. and a M.Sc. Degrees with Honors in Computer Engineering from Universidad Politécnica de Madrid. He holds a Ph.D. in Software, Systems and Computing from UPM School of Computer Science. His research interests include distributed systems, cloud computing and Internet of Things. Victor has published a number of papers on several international conferences.

Javier Soriano is Associate Professor of Computer Science at Universidad Politécnica de Madrid, UPM, Spain. He is Director of the Computer Networks and Web Technologies Laboratory (CETTICO Research Group). His research focuses on distributed systems and future Internet technologies. He holds a Ph.D. with Honors in Computer Science from UPM. Javier has leaded, as UPM Principal Researcher, a number of EU-funded international research projects including FAST, 4CaaST, MyMobileWeb, FI-WARE, FI-CORE and FI-NEXT. He has coauthored more than 60 papers published in international impact journals, research books and international conferences. Javier is a Senior Member of the IEEE since 2005.

David Lizcano holds a Ph. D. with honors in Computer Science (2010) from the Universidad Politécnica de Madrid, and a M.Sc. degree with honors in Research in Complex Software Development (2008) from the Universidad Politécnica de Madrid. He is Professor at Madrid Open University, UDIMA. He held a research grant from the European Social Fund, and involved in several national and European funded projects relating to Service Oriented Architectures, Paradigms of Programming, Software Engineering, Human-Computer Interaction and End-user Development. He has published his research in more than 25 prestigious journals indexed in relevant positions of the JCR.

Carlos Miguel holds a B.Sc. Degree in Computer Engineering from Universidad Politécnica de Madrid (UPM). He is student of the M.Sc. Degree in Computer Engineering and PhD candidate in Software, Systems and Computing from UPM. His research interests include distributed systems and cloud computing.