International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Jan-2022)

https://dx.doi.org/10.12785/ijcds/110108

Performance Factors Effect on the Performance Metrics of the Enterprise Service Bus

Lindung Parningotan Manik[1,2]

[1]*Research Center for Informatics, National Research and Innovation Agency, Bandung, Indonesia*
[2]*Faculty of Information Technology, University of Nusa Mandiri, Jakarta, Indonesia*

Received 26 Jan. 2021, Revised 21 Jul. 2021, Accepted 24 Jul. 2021, Published 9 Jan. 2022

Abstract: Enterprise Service Bus (ESB) is a modern software (or in the form of middleware) approach that implements Service-Oriented Architecture (SOA) and Enterprise Application Integration (EAI) principles. Independent services run and communicate with others through a high-level network protocol in a distributed computing environment. This research demonstrates a performance evaluation by comparing the selected performance metrics such as response time, throughput, and resource (CPU and memory) utilization on the chosen ESB products. Unfortunately, this type of study is very limited, and its publications are hard to find. Thus, our work could become a new framework reference for organizations when evaluating and choosing available ESB products in the market nowadays that fit organization needs. Furthermore, this research also studied the effect of chosen performance factors on the selected performance metrics through systematic measurements.

Keywords: Enterprise Service Bus, Performance evaluation, Middleware, Service-Oriented Architecture, Distributed computing

1. Introduction

Nowadays, business needs require enterprises to enable efficient and practical information and technology communication within their internal and external applications to achieve organizational goals. A common challenge faced by an organization is integrating all existing software (as well as legacy applications) with new internal or external software due to information technology complexity that comes from heterogenic applications with different programming languages and runs on different platforms. Enterprise Service Bus (ESB) [1], as one of the middleware infrastructure platforms, came to the rescue to tackle this problem. It provides easy integration of services and enterprise applications for complex architectures. It uses enterprise application integration [2] and service-oriented architecture [3] principles.

Enterprise Application Integration (EAI) is a framework that uses software (or in the form of middleware) and computer systems' architectural principles. It consists of various technologies and application services to integrate cross-corporate systems and enterprise computing applications. EAI can create various applications within an organization, including the system of partners to communicate with each other in achieving business objectives regardless of the platform and geographical location of the applications. EAI components consist of message transportation, message transformation, message routing, message logging, and business process management. In addition, EAI can be appropriately implemented in a distributed system by using various architecture such as the point-to-point, hub-and-spoke (client-server), multicast, publish-subscribe, or bus architecture [4].

On the other hand, Service-Oriented Architecture (SOA) integrates distributed software components or applications through a communication protocol over a high-level network. Today, SOA has another variant that we call microservices architecture which consists of a collection of small, autonomous services [5]. It is implemented in various applications such as in a smart city [6], a digital factory [7], tourism websites to support the electronic promotion and marketing [8], or in a cognitive media platform [9].

Apart from SOA, companies also can use traditional ways to integrate applications like presentation integration (screen scrapping), remote procedure invocation, data integration mechanisms such as data replication, shared database, file transfer, et cetera. The ESB platform wraps these methods. In addition, the ESB is an infrastructure for innovation [10] with a modular and concurrent design that can be used to accommodate SOA and EAI principles such as messaging routing, message transportation, logging, and protocol transformation.

E-mail address: lind008@brin.go.id, lindung.lpm@nusamandiri.ac.id

Bus architecture that is used by the ESB analogs to the bus concept found in computer hardware architecture. It uses a centralized message bus for message propagation. Applications may send messages to the bus by using adapters. These messages flow to the receiving application using the message bus. The receiving application has adapters that retrieve messages from the bus and translate the messages into the format required by the application. As the adapters have an integration engine and run on the same platform as the application source and designated application, scalability is no longer a problem. However, it is managed more complexly than the hub-and-spoke topology.

Internet of Things (IoT) and Big Data are popular technologies in recent years. The ESB architecture is also can be used as a middleware approach to accommodate the Internet of Things (IoT) needs [11] [12], big data processing [13] [14], and grid computing [15] as well. The software provides services that enable other machines to consume the services to interact with each other. This technology serves interoperability in supporting the coherent distributed architectural motion, where the middleware is often used to support and simplify the complexity of distributed applications.

In order to meet the requirement of a modern system that executes more complex cases in multiple platforms while delivering high Quality of Service (QoS) with high performance and high throughput, there must be a mechanism to evaluate the ESB performances. Ref [16] and [17] conducted surveys on several metrics such as management and security metrics on comparing existing ESBs in the market. The results showed that each ESB has its strengths and weaknesses. Nevertheless, the comparisons did not too detail in performance metrics. Furthermore, it is found in a recent systematic literature review that only 4% of the ESB studies in 2011-2019 conducted performance analysis research [18]. Therefore, this work was conducted to give new insights into ESB performance analysis as suggested in the previous studies.

This study presents novel technical results of an ESB performance evaluation obtained through measurements. This research also details the performance factors' effect on the selected performance metrics when comparing the ESB products as a step further from the previous studies. The ServiceMix and the Mule were selected in this study as the ESB products. As the limitation of the research scope, the system parameters kept constant were the server hardware, operating system, and java runtime environment. Therefore, the results can be used as a frame of reference for organizations on selecting performance parameters when comparing or evaluating and choosing one of the available ESB software products in the market.

The rest of this paper is structured as follows. First, in Section 2, limited related works are presented. Then, the performance evaluation methods are described in section Section 3, while the results and discussions are explained in section Section 4. Lastly, the conclusion is given in Section 5.

2. Related Work

As explained in Section 1, similar studies that conducted performance analysis of ESB are very limited. Ref [19] evaluated three ESB products: the Mule, the WSO2, and the ServiceMix. Meanwhile, the mean response time and the throughput were chosen as the performance metrics. As a result, the ServiceMix and the WSO2 were better than the Mule based on the computed throughput. However, the ServiceMix was better than the WSO2 based on computed response time.

In the other study, three ESB products, such as the WSO2, the Mule, and the Talend, were assessed [20]. Three performance metrics like the response time, the throughput, and the CPU utilization were evaluated. As a result, the Talend achieved the shortest response time, the Mule performed worst regarding the throughput values while the Talend and the WSO2 performance were close, and the Talend put the greatest load on the CPU while the WSO2 did not put an excessive load.

Ref [21] performed a simpler evaluation where the response time was selected as the only performance metric. Three ESB products, such as the Fuse, the Mule, and the Petals, were compared. As a result, the Mule achieved the lowest response time without the security service. Nevertheless, the Mule obtained the highest response time with the security service.

3. Methods

In this section, the overview of system performance evaluation and the chosen technique is briefly explained. The methodology used in this evaluation study, which is divided into ten steps [22], can be seen in Figure 1. The first step in the performance evaluation is setting the objectives and defining the systems as well as its limitation. As explained in Section 1, the objective of this study is to evaluate the magnitude of performance factors' effect on the performance metrics of the ESB. The system under evaluation in this case study was an internet payment gateway (IPG). The system provided various transaction processing services. First, the system design is explained in [23]. This design was then implemented in two ESB products mentioned before, i.e., the ServiceMix and the Mule.

An IPG is an instant payment service where users can make a payment transaction with an e-wallet balance or debit/credit card for products purchased by the payment provider's partners or merchants. However, this explanation is simplified since many other use-cases are not covered in this research. The services owned by the middleware system served as a gateway between the internal and the external systems. System architecture displayed in Figure 2 shows

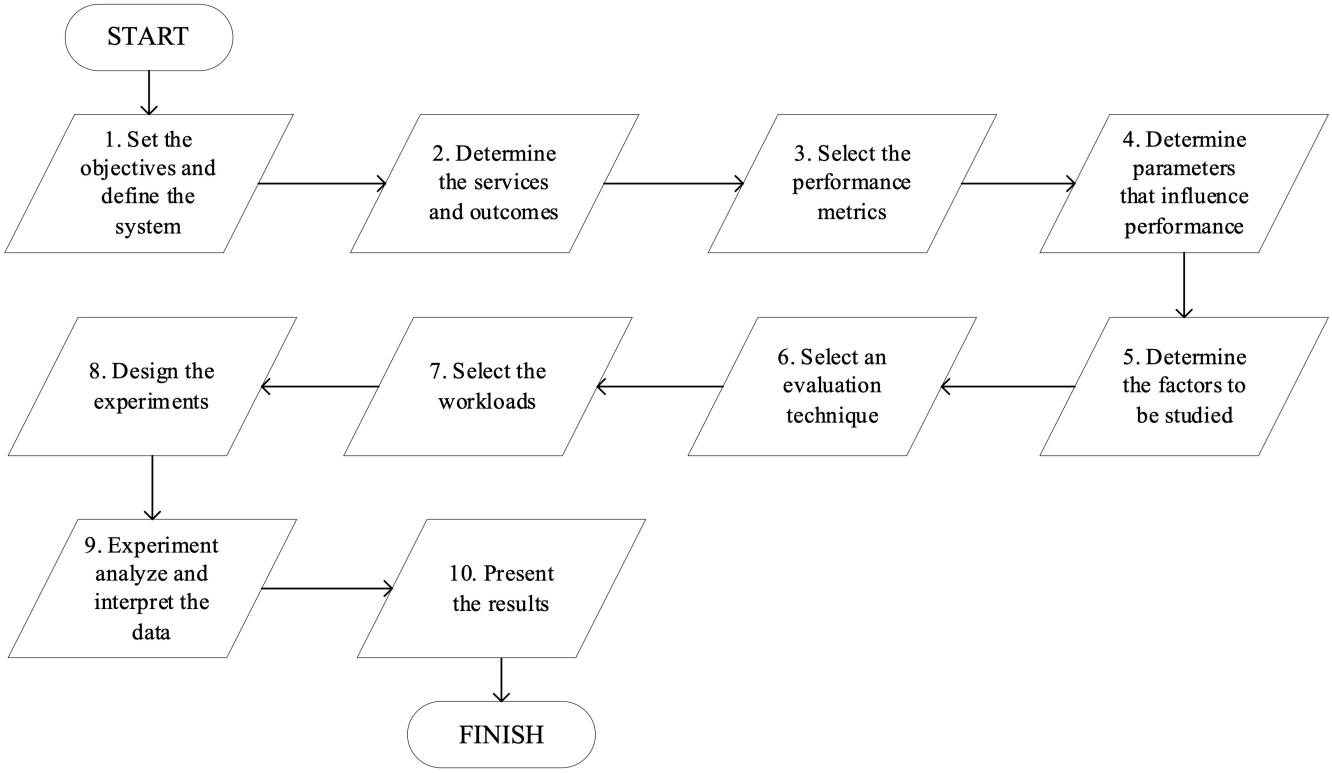

Figure 1. Research Methods

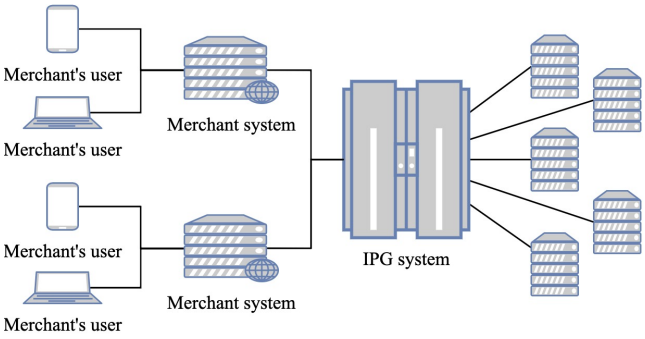

Figure 2. System architecture

the communication that the systems make. First, when users perform transactions, the requests are forwarded to the IPG system. Next, middleware (ESB) inside the system process the transactions by forwarding them to the relevant internal or external systems. Finally, responses obtained from the systems are processed and forwarded back to the users.

After defining the system and determining the services, the next step in analyzing and evaluating the system is selecting the performance metrics, the parameters that influence performance, the factors to be studied, the evaluation technique, the workloads, et cetera. In order to make a structured explanation, these steps are described in the following sections. Step three until step five is defined in Section 3-A. Step six until step eight is explained in Section 3-B, 3-C, and 3-D, respectively. Lastly, steps nine and ten are described in Section 4.

A. *Metrics, Parameters, and Factors of Performance*

As mentioned before, the next step is to select the criteria for performance comparison. These criteria are also called the metrics. In general, the metric is associated with the speed, accuracy, and availability of services. If the system delivers the service properly, then the performance is measured by the time spent to deliver the service, the speed of service delivered, and the resources consumed while delivering the service. The three metrics are related to the time-speed-resources for successful performance or responsiveness, productivity, and utilization metrics. Due to limited resources, the performance evaluation was executed only limited to services that can be adequately delivered, while the errors and failures were ignored. Therefore, the performance metrics used were:

1) Response time
   It is defined as the interval between a user request and the response of the system. This definition, however, is simplified because the request or response does not occur instantaneously. In a database system, response time is defined as the user's time from querying the database with an input key until getting a result value [24]. In this case study, the user spends time typing a request while the system returns

a response. There are two possible definitions of response time for this case. The response time can be defined as the interval between the end of submitting a request and the initial response correspondence of the system or the interval between the end of submitting a request and the end of response correspondence of the system. In this study, the definition used is the second one.

2) Throughput
It is defined as a request per time unit. In a web application, throughput is defined as how many requests the system can be served in a second [25]. Since the system under evaluation was a transaction processing system, its throughput was measured in Transactions Per Second (TPS). It is assumed that C_i as the number of successfully processed transactions during the time T, then the throughput, X_i, can be calculated by the following equation.

$$X_i = \frac{C_i}{T} \quad (1)$$

3) Resource utilization (RU)
It is measured by the fraction of the busy time of resources in serving a request. So, its value is the ratio of busy time to total time during a specified period.

$$RU = \frac{\sum resource's\ busy\ time}{\sum measurement\ duration} \quad (2)$$

4) System availability
It is defined by the system's time until the system is available to serve user requests.

The benchmarking process was carried out by comparing these performance metrics between the systems that were evaluated. After defining the performance metrics, the next step is to determine the parameters that influence performance. These parameters are divided into the system parameters, such as hardware or software, and the workload parameters, which vary users' requests. Therefore, the parameters' values can be varied or kept constant during performance evaluation. The influential parameters in this evaluation were:

1) System parameters
   a) The hardware used
   It included a server that contains CPU, memory, disk, and network. These parameters were a constant value.
   b) The software used
   It included the ESB product, message broker, database, and operating system. The parameters of the message broker, database, and operating system were a constant value.
2) Workload
This parameter changed in value. The explanation is given in Section 3-C.

The parameters that have varied values, such as ESB product and workload, are also called the performance factors, and their values are called levels. As mentioned in the Section 1, the values of ESB products were the Mule and the ServiceMix, while the detailed explanation of the workloads is given in Section 3-C. The performance metrics above are classified into three categories. They are:

1) HB, Higher is Better
The performance metric included in this category is throughput (Transactions per second).
2) LB, Lower is Better
The performance metrics included in this category are response time, availability time, and resource utilization (CPU and memory usage).
3) NB, Nominal is Best
Both high and low values are equally undesired in this category, the medium value is the best. Typically, resource utilization falls into this category since a high value is not desired. However, a low value is also not desired because this indicates a significant amount of hardware system resources that are not utilized. A value in the range of 50% -75% may be considered the best. Nevertheless, since the hardware parameter's value was kept constant, it would be inappropriate to include resource utilization in this category. Therefore, the use of resources by the middleware system is included in the LB category, where the use of fewer resources is better than higher ones.

B. Evaluation Technique

Three techniques can be used for performance evaluation, i.e., analytical modeling, simulation, and real system measurement [26]. The selection of the correct technique is based on the time and resources available to solve the problem and the desired level of accuracy.

TABLE I. Comparison of Evaluation Techniques

Criteria	Analytical Modelling	Simulation	Measurement
Stage	-	-	Postprototype
Required time	Short	Medium	Varies
Tools	Analysis	Computer language	Instrumentation
Accuracy	Low	Medium	Varies
Evaluation of results	Easy	Medium	Hard
Cost	Low	Medium	High
Saleability	Low	Medium	Intermediate

In Table I, the comparisons of the evaluation techniques are shown. All techniques have their own advantages and disadvantages. However, since the system under evaluation had been deployed in a production environment, the actual

measurement was selected as the performance evaluation technique.

C. Workload

The workload consists of a list of services that make requests to the system. Based on the selection of evaluation techniques, chosen workloads can be expressed in different formats. It is important to note that the workload should represent the use case of a natural system in all cases. In order to produce a representative workload, it is necessary to measure and represent the current system. There are four primary considerations in selecting workloads: the service delivered by the workload, the level of detail, the level of representation, and the timeline.

Since the system under evaluation was a transaction processing service, the workload representation was the frequency of transactions. The workload was given by using the merchant simulator to the ESBs that were evaluated. In order to determine the response time and the transactions per second, the query to the middleware database was used to calculate the difference between the incoming request time and the outgoing response time. To measure the performance utilization during the test, the monitoring tools of the operating system, such as the performance monitor, were used.

Three workload parameters were chosen: the number of users' requests, the time interval between one request and another request, and service type. Since these parameters changed in value, they were also selected as the performance factors besides the ESB product. The number of users' requests was divided into two levels, which were 50 and 100 transactions. The values of the interval between one request and another request were 50 and 100 ms. Meanwhile, the level values in these factors were not based on the actual value, in this case, the statistical data in the existing production environment. However, the chosen values were based on the future forecast, so that it was deemed necessary to assign an extreme value to the conducted performance test. Finally, the service type value was also divided into two levels: payments using an e-wallet balance and a credit card.

D. Design of Experiments

After determining a list of factors and levels, it is necessary to determine the sequence of experiments that can yield much information with minimal effort. In practice, it is helpful if the experiment is divided into two phases. In the first phase, the number of factors is significant, but the number of levels is negligible. The objective is to determine the effects of varied factors. In the second phase, the number of factors is reduced, but the level value of factors having significant impacts increases. Since the measurement was selected as the performance evaluation technique in this study, the experiments must be designed accurately to generate maximum information with a minimum number of measurements. A precise analysis of the test results may separate which factors affect the performance significantly. Hence, the full factorial design was used by utilizing every possible combination at all levels of all factors. The formula to calculate the number of experiments, n, is defined in the following equation.

$$n = \prod_{i=1}^{k} n(i) \tag{3}$$

The advantage of this method is its capability of analyzing all possible configurations and workloads. The effect of each factor may be discovered, including secondary factors and how they interact with each other. The main problem of this method is its cost of study. This method requires a lot of time and money to conduct each test, particularly when each test should be repeated several times. There are three ways to reduce the number of tests, namely:

1) Reducing the number of levels for each factor.
2) Reducing the number of factors.
3) Using fractional factorial designs.

The first alternative is the most recommended. In some cases, each factor may use two levels only to determine the importance of these factors. A full factorial design where each factor, k, has two levels is also referred to as the 2^k design. This design requires experiments in a number of 2^k. By using this method, it is easy to analyze the impact of the factors. However, one of the problems with this design is the impossibility of estimating the test of errors as there is no repeat test. Therefore, the $2^k r$ design is needed where r is the number of tests. To determine the average of the results, then the mean, $\bar{x}$, is calculated by the following equation.

$$\bar{x} = \frac{1}{r} \sum_{i=1}^{r} x(i) \tag{4}$$

E. Performance Test

As explained in Section 3-A and 3-C, there were four performance factors in this evaluation study. Each of these factors was only allowed to have two levels, and the number of repeat tests was set to three times. Therefore, the number of experiments required was $2^4 \times 3 = (16 \times 3)$, i.e., 48 tests. For the factor of service type, only two levels were selected: payment using e-wallet balance (Service-1) and payment using credit card (Service-2). In addition, ESB products are specified in Section 1, which are the ServiceMix and the Mule.

As also mentioned in Section 3-C, the number of transaction requests was set to 50 and 100 transactions. Meanwhile, the levels of the time interval between requests were set to 50 and 100 ms. The performance factors and levels are shown in Table II, while the test cases performed in each ESB are shown in Table III. The environment test

TABLE II. The Performance Factors and Levels

Performance Factors	Level -1	Level 1
Service Type, A	Service-1	Service-2
Time interval between requests, B	50 ms	100 ms
Number of transaction requests, C	50	100
ESB product, D	ServiceMix	Mule

TABLE III. The Performance Test Cases

No	Test case
Scenario 1	A = Service-1 B = 50 ms C = 50 transactions
Scenario 2	A = Service-1 B = 50 ms C = 100 transactions
Scenario 3	A = Service-1 B = 100 ms C = 50 transactions
Scenario 4	A = Service-1 B = 100 ms C = 100 transactions
Scenario 5	A = Service-2 B = 50 ms C = 50 transactions
Scenario 6	A = Service-2 B = 50 ms C = 100 transactions
Scenario 7	A = Service-2 B = 100 ms C = 50 transactions
Scenario 8	A = Service-2 B = 100 ms C = 100 transactions

used was as follows:

1) Hardware server: HP Proliant ML 350 G3.
   a) Intel Xeon 3.06 GHz
   b) 512 KB L2 Cache
   c) FSB 533 MHz
   d) 1 x 2GB RAM
2) Windows Server 2003 Enterprise Edition Service Pack 2
3) Sun Java Development Kit 1.6 update 6

4. Results and Discussions

In this section, the results of the experiments and the comprehensive discussion are given.

A. Results

From the test results, the comparison of performance metrics between the design of middleware implemented on ServiceMix and Mule are analyzed as follows:

1) Availability time
   In terms of the availability time which is time interval between the system started until the system became available for serving requests, the middleware implemented on the ServiceMix was faster than the Mule. On average, the ServiceMix took 22918 ms while the Mule took 25431.8 ms. Since this metric belongs to the LB category, where a lower value is better than a higher value, the ServiceMix was superior to the Mule.
2) Response time
   The comparison of response time between the ServiceMix and the Mule can be seen in Figure 3, where the x-axis is the test scenario number, while the y-axis is the mean of the response time and S is the test conducted on the ServiceMix, while M is the test conducted on the Mule. The average value of response time on the ServiceMix was lower than the Mule in almost all tests. Since this metric belongs to the LB category, where a lower value is better than a higher value, the ServiceMix was superior to the Mule.
3) Throughput
   The comparison of throughput between the ServiceMix and the Mule can be seen in Figure 4, where the x-axis is the test scenario number, while the y-axis is the mean of the throughput and S is the test conducted on the ServiceMix, while M is the test conducted on the Mule. The average value of throughput on the ServiceMix was more higher than the Mule in almost all tests. Since this metric belongs to the HB category, where a higher value is better than the lower value, the ServiceMix was superior to the Mule.
4) Resource utilization
   The comparison of resource utilization (CPU and memory usage) can be seen in Figure 5 and Figure 6, where the x-axis is the test scenario number, while the y-axis is the mean resource utilization and S is the test conducted on the ServiceMix, while M is the test conducted on the Mule. The average value of resource utilization in either processor usage or memory usage on the ServiceMix was more higher than the Mule in almost all tests. Since this metric belongs to the LB category where a lower value is better than a higher value, the Mule was superior to the ServiceMix.

B. Discussions

In order to find out the magnitude of performance factor, k, effects on the performance metrics, the total variation SST (Sum of Squares Total), is calculated beforehand. From the test results, to obtain the value of SST, the $2^k \times 2^k$ matrix (sign table as shown in Table IV) is made between the performance factors and the levels, where Y_1, Y_2, ..., Y_{16} are the experiment results respectively on each performance factors. The total variation explained by the effect of all performance factors, which includes k

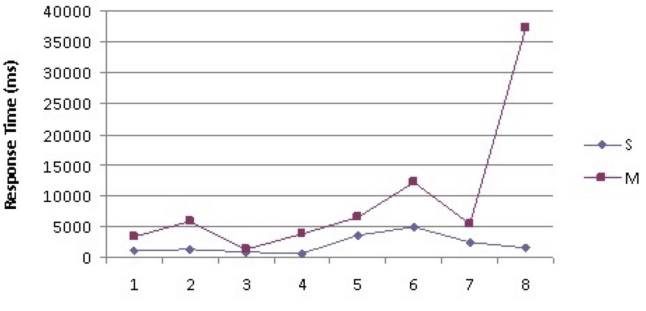

Figure 3. Response time performance metric

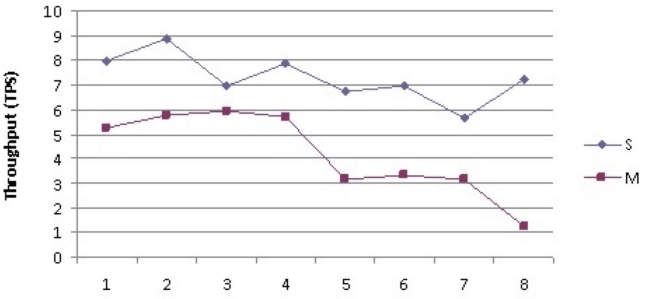

Figure 4. Throughput performance metric

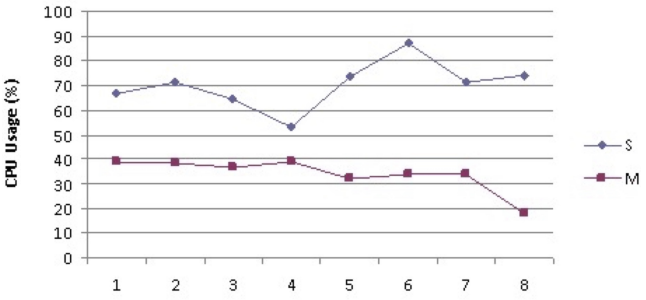

Figure 5. CPU usage performance metric

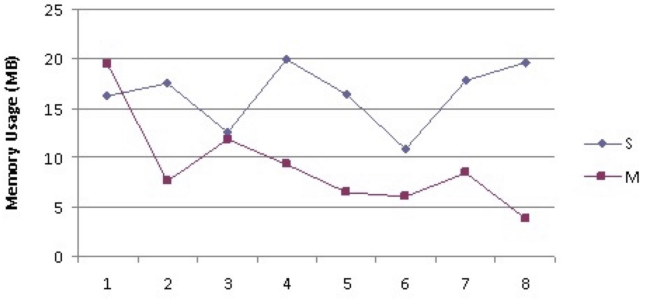

Figure 6. Memory usage performance metric

main effects and also the interactions between the factors, $I \in \{A, B, C, D, AB, AC, AD, BC, BD, CD, ABC, ..., ABCD\}$, can be calculated by the following equation.

$$\begin{aligned} SST = 2^4(&SS_A{}^2 + SS_B{}^2 + SS_C{}^2 + SS_D{}^2 \\ &+SS_{AB}{}^2 + SS_{AC}{}^2 + SS_{AD}{}^2 \\ &+SS_{BC}{}^2 + SS_{BD}{}^2 + SS_{CD}{}^2 \\ &+SS_{ABC}{}^2 + SS_{ABD}{}^2 + SS_{ACD}{}^2 + SS_{BCD}{}^2 \\ &+SS_{ABCD}{}^2) \end{aligned} \tag{5}$$

Meanwhile, the importance of factor I is measured by the proportion vp_I of the total variation explained by the factor. Important factors are those that explain a large percentage of variation. It can be expressed as a fraction of sum of squares due to I, SS_I, to the SST. The percentage of variation helps the researchers in determining if it is worthwhile to study a factor or interaction further. The SS_I and vp_I can be calculated by the following equations.

$$SS_I = \frac{\sum_{j=1}^{2^k} I_j Y_j}{2^k} \tag{6}$$

$$vp_I = \frac{SS_I{}^2}{SST} \tag{7}$$

As can be inferred from Table V, it is found that the chosen ESB product was the most influential factor among others on the response time metric. Another factor that was also quite decisive in this metric was the type of service. Meanwhile, other factors, such as the interaction between the four factors, were influential on this metric and memory usage metric, which was around 40-55%. It can also be found that the chosen ESB product was the most influential factor among others on the throughput and resource utilization metric. Similarly, like the response time metric, another factor that was also quite decisive in the throughput and memory usage metric was the type of service carried out.

Meanwhile, other factors, such as the interaction between the four factors, had little effect on the throughput and CPU usage metric, around 12-13%. The time interval between requests was the most insignificant factor in the metrics. Even though the number of transaction requests factor had little effect on the response time metric, it was also an insignificant factor to the other metrics.

5. Conclusions

A novel technical result in ESB performance evaluation through a systematic measurement of various experiments is presented in this paper. The effect of performance factors on the metrics of the ESB has also been studied. In conclusion, the most affecting factor was the ESB product used in

TABLE IV. SST Sign Table

I	A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	ABCD	Y
1	-1	-1	-1	-1	1	1	1	1	1	1	-1	-1	-1	-1	1	Y_1
2	1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	-1	-1	Y_2
3	-1	1	-1	-1	-1	1	1	-1	-1	1	1	1	-1	1	-1	Y_3
4	1	1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	1	1	1	Y_4
5	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	Y_5
6	1	-1	1	-1	-1	1	-1	-1	1	-1	-1	1	-1	1	1	Y_6
7	-1	1	1	-1	-1	-1	1	1	-1	-1	-1	1	1	-1	1	Y_7
8	1	1	1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1	-1	Y_8
9	-1	-1	-1	1	1	1	-1	1	-1	-1	-1	1	1	1	-1	Y_9
10	1	-1	-1	1	-1	-1	1	1	-1	-1	1	-1	-1	1	1	Y_{10}
11	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	Y_{11}
12	1	1	-1	1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1	Y_{12}
13	-1	-1	1	1	1	-1	-1	-1	-1	1	1	1	-1	-1	1	Y_{13}
14	1	-1	1	1	-1	1	1	-1	-1	1	-1	-1	1	-1	-1	Y_{14}
15	-1	1	1	1	-1	-1	-1	1	1	1	-1	-1	-1	1	-1	Y_{15}
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Y_{16}
																Total

TABLE V. The Proportion of Factor Variation to The Performance Metric

	The Proportion of Factor Variation to (%)			
Performance Factor	Response time	Throughput	CPU Usage	Memory Usage
A	16.464	27.167	0.238	8.832
B	1.106	1.735	2.724	0.099
C	9.649	0.432	0.432	0.432
D	18.489	58.501	84.219	84.219

the evaluation compared to other selected factors like the type of service, the time interval between requests, and the number of transaction requests. Since the effect of the ESB product factor was the greatest amongst other factors, therefore organizations must conduct other tests using more ESB software products when choosing the best available in the market. It should be done to obtain the best selection using the ratio of cost (either the cost incurred, or the time needed for training, research, and development) to the performance. The greatest ratio may be selected as the best.

The performance evaluation could be conducted more appropriately if the number of performance factors and levels is increased. For example, to determine the purchase of hardware specifications (such as processor, memory, cache, and so on), it is necessary to conduct tests by considering these factors with varying levels. Furthermore, the performance evaluation results could be better if other evaluation techniques are performed, for example, with the analytic modeling using the queuing theory to complete the test conducted before, i.e., by measurement. In addition to cutting costs, the analytical modeling may also validate the measurement results and predict the resulting performance if the levels of factor determined are numerous.

Acknowledgment

The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen, "The enterprise service bus: Making service-oriented architecture real," *IBM Systems Journal*, vol. 44, no. 4, pp. 781–797, 2005.

[2] Z. Irani, M. Themistocleous, and P. E. Love, "The impact of enterprise application integration on information system lifecycles," *Information & Management*, vol. 41, no. 2, pp. 177–187, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378720603000466

[3] N. Niknejad, W. Ismail, I. Ghani, B. Nazari, M. Bahari, and A. R. B. C. Hussin, "Understanding service-oriented architecture (soa): A systematic literature review and directions for further investigation," *Information Systems*, vol. 91, p. 101491, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437920300028

[4] S. J. Green, "An evaluation of four patterns of interaction for integrating disparate esbs effectively and easily," *Journal of Systems Integration*, vol. 4, pp. 3–19, 07 2013.

[5] N. Nagarajan and R. Thirunavukarasu, "Service-oriented broker for effective provisioning of cloud services – a survey," *International Journal of Computing and Digital Systems*, vol. 9, no. 5, pp. 863–879, Sep. 2020. [Online]. Available: https://doi.org/10.12785/ijcds/090508

[6] C. Badii, P. Bellini, A. Difino, P. Nesi, G. Pantaleo, and M. Paolucci, "Microservices suite for smart city applications," *Sensors*, vol. 19, no. 21, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/21/4798

[7] M. Ciavotta, M. Alge, S. Menato, D. Rovere, and P. Pedrazzoli, "A microservice-based middleware for the digital factory," *Procedia Manufacturing*, vol. 11, pp. 931–938, 2017, 27th

International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978917304055

[8] S. Adi, "A collaborative tourism websites to support electronic promotion and marketing by using enterprise service bus (ESB)," *Advanced Science Letters*, vol. 22, no. 5, pp. 1179–1183, May 2016. [Online]. Available: https://doi.org/10.1166/asl.2016.6735

[9] P. B. Putera, L. P. Manik, Y. Rianto, A. A. Sari, and R. Sadikin, "How indonesia uses big data "indonesian one data" for the future of policy making," *International Journal of Advanced Science and Technology*, vol. 29, no. 05, pp. 2177 – 2185, Apr. 2020. [Online]. Available: http://sersc.org/journals/index.php/IJAST/article/view/10980

[10] B. Bygstad and H.-P. Aanby, "ICT infrastructure for innovation: A case study of the enterprise service bus approach," *Information Systems Frontiers*, vol. 12, no. 3, pp. 257–265, May 2009. [Online]. Available: https://doi.org/10.1007/s10796-009-9169-9

[11] F. Wang, L. Hu, J. Zhou, and K. Zhao, "A data processing middleware based on SOA for the internet of things," *Journal of Sensors*, vol. 2015, pp. 1–8, 2015. [Online]. Available: https://doi.org/10.1155/2015/827045

[12] B. Negash, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "LISA 2.0: lightweight internet of things service bus architecture using node centric networking," *Journal of Ambient Intelligence and Humanized Computing*, vol. 7, no. 3, pp. 305–319, Mar. 2016. [Online]. Available: https://doi.org/10.1007/s12652-016-0359-2

[13] C. Orłowski, E. Szczerbicki, and J. Grabowski, "Enterprise service bus architecture for the big data systems," *Management and Production Engineering Review*, no. No 1, 2014. [Online]. Available: http://journals.pan.pl/Content/89479/PDF/4-orlowski.pdf

[14] T. Górski, "The use of enterprise service bus to transfer large volumes of data," *Journal of Theoretical and Applied Computer Science*, vol. 8, no. 4, pp. 72–81, 2014.

[15] Riad, "Design of SOA-based grid computing with enterprise service bus," *International Journal on Advances in Information Sciences and Service Sciences*, 2010. [Online]. Available: https://doi.org/10.4156/aiss.vol1.issue1.6

[16] R. S. Bhadoria, N. S. Chaudhari, and G. S. Tomar, "The performance metric for enterprise service bus (esb) in soa system: Theoretical underpinnings and empirical illustrations for information processing," *Information Systems*, vol. 65, pp. 158–171, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437915301952

[17] M. Martínez-Carreras, F. G. Jimenez, and A. G. Skarmeta, "Building integrated business environments: analysing open-source esb," *Enterprise Information Systems*, vol. 9, no. 4, pp. 401–435, 2015. [Online]. Available: https://doi.org/10.1080/17517575.2013.830339

[18] O. Aziz, M. S. Farooq, A. Abid, R. Saher, and N. Aslam, "Research trends in enterprise service bus (esb) applications: A systematic mapping study," *IEEE Access*, vol. 8, pp. 31 180–31 197, 2020.

[19] S. P. Ahuja and A. Patel, "Enterprise service bus: A performance evaluation," *Communications and Network*, vol. 03, no. 03, pp. 133–140, 2011. [Online]. Available: https://doi.org/10.4236/cn.2011.33016

[20] T. Górski and K. Pietrasik, "Performance analysis of enterprise service buses," *Journal of Theoretical and Applied Computer Science*, vol. 10, pp. 16–32, 10 2017.

[21] F. García-Jiménez, M. Martínez-Carreras, and A. Gómez-Skarmeta, "Evaluating open source enterprise service bus," in *2010 IEEE 7th International Conference on E-Business Engineering*, 2010, pp. 284–291.

[22] R. Jain, *Art of Computer Systems Performance Analysis: Techniques For Experimental Design Measurements Simulation and Modeling*. Wiley, 2015.

[23] L. P. Manik, "Design pattern evaluation on a restful api wrapper: A case study of software integration with an internet payment gateway using model-driven architecture," *Journal of Information Technology and Computer Science*, vol. 4, no. 3, p. 222–232, Dec. 2019. [Online]. Available: https://jitecs.ub.ac.id/index.php/jitecs/article/view/107

[24] P. Pirzadeh, J. Tatemura, O. Po, and H. Hacıgümüş, "Performance evaluation of range queries in key value stores," *Journal of Grid Computing*, vol. 10, no. 1, pp. 109–132, Mar. 2012. [Online]. Available: https://doi.org/10.1007/s10723-012-9214-7

[25] A. Yates, K. Beal, S. Keenan, W. McLaren, M. Pignatelli, G. R. S. Ritchie, M. Ruffier, K. Taylor, A. Vullo, and P. Flicek, "The ensembl REST API: Ensembl data for any language," *Bioinformatics*, vol. 31, no. 1, pp. 143–145, Sep. 2014. [Online]. Available: https://doi.org/10.1093/bioinformatics/btu613

[26] M. R. Galankashi, E. Fallahiarezoudar, A. Moazzami, N. M. Yusof, and S. A. Helmi, "Performance evaluation of a petrol station queuing system: A simulation-based design of experiments study," *Advances in Engineering Software*, vol. 92, pp. 15–26, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096599781500143X

Lindung Parningotan Manik Lindung Parningotan Manik currently works as a Researcher at the Research Center for Informatics, National Research and Innovation Agency (BRIN), and a Lecturer at the Computer Science Graduate Program, University of Nusa Mandiri, Indonesia. He received the Professional Doctorate in Engineering (PDEng) degree from Technische Universiteit Eindhoven, in 2015. He has extensive experience in software engineering. Since 2008, he has working as a Software Designer in multiple (high) tech industries. His research interests include software technology, data mining, and information system engineering.