ДОДАТОК А

Апробація результатів роботи

Problems of infocommunications.

Science and technology

IEEE International Scientific and Practical Conference

PIC S&T′2019

Київ

8 – 11 Жовтня 2019

305

The Relevance of Using Message Brokers in Robust Enterprise Applications

Vladyslav Apukhtin, Mariya Shirokopetleva, Victoria Skovorodnikova
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
vladyslav.apukhtin@nure.ua, marija.shirokopetleva@nure.ua viktoriia.skovorodnikova@nure.ua

***Abstract*–The paper covers main message broker protocols and the patterns which are used to manage technical problems. The article provides the reader with typical software engineering design issues which might lead to conceptual violations, software non-functional requirements nonconformity, which is high coupling problem, synchronous operation performance loss, and system constituent integration problem. The solutions which come as a result of applying messaging are brought covering each problem above, in particular establishing the communication between loosely coupled applications, researching the application performance speedup with the Amdahl's law and applying Read Model architectural pattern leveraging message brokers capabilities.**

Keywords – messaging, asynchronous operations, Amdahl's law, protocol, application integration.

I. Introduction

Enterprise applications are applications that display, manipulate, and store large amounts of data and support or automate business processes with that data [1]. Such applications usually consist of separated modules, services or even clusters of services that handle business tasks differently. Taking into account the complexity of the architecture, there is a requirement to support integration between the components of the system. The idea of message brokers has come about as an adequate and flexible solution for this purpose.

The message brokers are a good instrument for resolving typical computer system problems. Section II describes the main messaging concepts and addresses the papers of the scientific field covered. Section III presents which issues can arise during the application development process. The practical solutions of them are presented in section IV which are aimed to solve distributed services communication ones in addition to centralizing distributed data via the messaging capabilities.

II. Literature Review

The Message broker is an intermediate software that is responsible for sending the messages of a format defined between software components using the specific protocol, commonly MQTT (Message Queuing Telemetry Transport), STOMP (Streaming Text Oriented Messaging Protocol), AMQP (Advanced Message Queuing Protocol).

Messaging systems typically involve the introduction of an intermediary between the two systems that communicate in order to further decouple the sender from the receiver or receivers. In doing so, the messaging system allows a sender to send a message without being aware of receiver, whether it is active, or indeed how many instances of them there are [3]. Different techniques (in other words – patterns) exist to achieve this goal, namely:

- Publisher-subscriber.
- Exclusive point-to-point.
- RPC (Remote procedure call).

The message is the amount of data that is considered to be exchanged. Usually, messages are restricted in size. Besides, there are different messaging publishing modes:

- *Exactly-once* i.e. the message can be delivered only once.
- *At-least-once* i.e. the message can be delivered multiple times with duplications.

Messages can be persistent on non-persistent which denotes that after message broker's breakdown there is the technical opportunity to redeliver messages or not respectively.

Message can contain properties and headers. Message properties are specific message characteristics, for instance, priority or content type.

Headers are message characteristics that can be flexibly changed. In opposite to message properties, headers can be taken into account by a broker to resolve the recipient.

Publisher-subscriber pattern brings the concepts of sender, receiver, message, and topic.

Publisher and Subscriber are the nodes of communication that exist in the many-to-many relationship; the nodes are responsible for sending and receiving messages respectively using exchange topics.

Exchange topics are string values that are used by the message broker to identify which subscriber is intended to receive a message that has been sent. The publisher sends a message to the broker with the exchange topic, the message broker resolves which consumers are capable of receiving the message.

Exchange topic key can be specified by consumer either as a direct value of the topic's name or following to the pattern.

It is assumed that a single message can be delivered to multiple receivers.

978-1-7281-4184-8/19/$31.00 ©2019 IEEE

The *exclusive point-to-point pattern*s used in case the publisher needs to send a message to only one message consumer. Even though multiple consumers may be listening on the queue for the same message, only one of those consumer threads will receive the message [4].

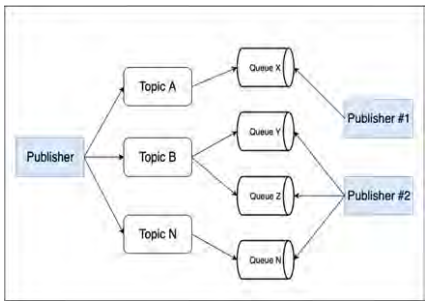


Fig. 1. Publisher-subscriber pattern typical infrastructure

The single message is delivered to exactly one consumer, the order of messages remains consistent.

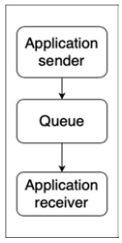


Fig. 2. Exclusive point-to-point pattern typical infrastructure

RPC pattern is an extension of the exclusive point-to-point pattern. As opposite to other mentioned messaging pattern, two-way communication exists between sender and receiver. For this purpose, usually two different channels are created, specifically for request and reply messages which are treated as a single request using the identifier – the correlation id.

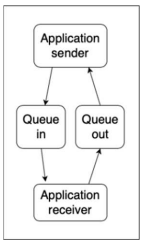


Fig. 3. RPC pattern typical infrastructure

Message brokers are actively used both in a single computer system environment and distributed one implementing different patterns. Jun-Young Byun at el.[5] proposed a DMBS cloud ecosystem with the underlying publisher-subscribe pattern. The deployed infrastructure can rapidly process messages enhancing the message response speed within multiple regions. Despite the extensive messaging service has been built up, the data was separated and accessing cross-cutting data implies additional resource expenses.

As the article investigates the asynchronous processing using the RPC pattern, the proposed research of el.[6] have been taken into account. The combination of Grizzly framework for the HTTP server and Jersey server has been taken as a target of the review. However, much of the work depended more on the programmatic Java application solutions, particularly depending on the Java thread pool limitations rather than focusing on moving parallel computing to the message broker instances. Therefore, end-to-end latency measurements might have had different results.

A. MQTT protocol

MQTT protocol (which stands for Message Queuing Telemetry Transport) is a lightweight real-time messaging protocol which leverages Publisher-Subscriber pattern to deliver messages in binary format using underlying TCP/IP protocol.

The lightness of the protocol is achieved by low packet size overhead which varies from 1 byte to 256 megabytes. In addition, implementations are designed to support low power usage. The protocol is also designed so that the implementation should consider limited network bandwidth and high latency.

Taking into account abovementioned, the protocol is commonly used in communication between smartphones, IoT (Internet of Things) devices, routers, etc.

The implementations of the MQTT protocol include but is not limited to Eclipse Mosquitto, eqmtt, VerneMQ.

B. STOMP protocol

Streaming Text Oriented Messaging Protocol as its name suggests is a text-oriented messaging protocol. The protocol leverages the publisher-subscriber pattern and is designed to be simple to use. The topics are called destinations.

The protocol has plenty of implementations, for instance, ActiveMQ, RabbitMQ, Stampy.

C. AMQP protocol

AMQP (Advanced Message Queuing Protocol) is an application-level binary protocol that leverages underlying TCP (Transmission Control Protocol) to asynchronously stream messages across the network in the binary format.

Comparing with MQTT and STOMP, AMQP is quite extensive and provides a wide range of capabilities for applications.

Transmitted data can be encrypted using TLS (Transport Layer Security). The protocol supports consistency by implementing the features of delivery acknowledgment, atomicity in a single queue though without complete transaction support (For instance, RabbitMQ implementation of the protocol enables transaction support within message publish, acknowledgment, reject while resource creation/deletion such as queues and exchanges is not implemented to be transactional). AMQP also guarantees that messages that have been published in a specific order will be delivered to

consumers in strictly the same order despite the count of the consumers.

AMQP support all message publish modes which were mentioned earlier as well as the patterns.

Taking into account all of the above, this protocol is preferred to be used in enterprise applications.

A typical implementation is RabbitMQ.

III. Problem Formulation

In order to understand the reasons for introducing message brokers, the following concerns should be researched: component decoupling, a disadvantage of synchronous operations and the complexity of new components integration.

A. Component decoupling

The statement 'High cohesion, low coupling' is a principle of General Responsibility Assignment Software Patterns. The lower the strength between the system constituents, the more system is considered to be decoupled [2]. In terms of enterprise applications, it is represented in the following: code dependencies, remote invocation dependencies within the range of protocols (HTTP[S], TCP, SOAP, etc.)

Given a component A that references component B which itself references component C and A, and the following chain of calls should be performed in order to send an arbitrary message to component Z. Such an example structure is a typical example of a highly coupled system.

The problem is how to avoid code dependencies regardless of whether components belong to a single application or not.

B. The disadvantage of synchronous operations

Synchronous operation A is an operation that is executed by a program that does not allow to proceed with the program execution to operation B until A has been executed. In other words, synchronous operation blocks the current thread. The examples are I/O operations, HTTP[S] request.

Asynchronous operation is an operation that does not block the current thread.

Making time-consuming operations asynchronous, the application can be considered responsive. Such applications leverage parallelism or parallel computing.

Parallel computing is a type of computation where operations can be executed simultaneously (in parallel).

In terms of enterprise applications, the example of a long-term synchronous operation that takes a relatively significant amount of time is sending an email.

The problem is how to make synchronous operations viable regardless of whether components containing long-running operations belong to a single application or not?

C. The complexity of new component integration

Integration of a new component partially coincides with component decoupling. Enterprise applications rarely consist of a single executing instance. In contrary, specific standalone features that are not related to core ones, are usually developed in the different application instance. In this case, the architectural concern arises: how should new component response different kind of events happen in core application and vice versa?

Another integration problem lies in migration data from one database probably outdated and not supported to a different one. The issue takes place in real-time applications that should handle current data and have an opportunity to merge old ones live.

IV. Methods and Analysis

Messaging could be brought as an instrument for solving abovementioned problems.

Dealing with decoupling of the application components leads to referencing the message broker instead of the other components itself. Component A is no more responsible for determining which nodes are considered to be the receiver of a message, how to handle receiver breakdown, how much receivers exist, should receiver accept the message or not. Instead, the component simply pushes the message to the direct topic name or pattern topic name and a message broker delivers it to component B, C, etc. The topic listener components which can either subscribe or unsubscribe the topic define the dependency on receiving the message itself. The breakdown of a node listening to a message causes the broker to redeliver the message as many times as it is configured. Moreover, different channel topics distinguish the receivers of an event.

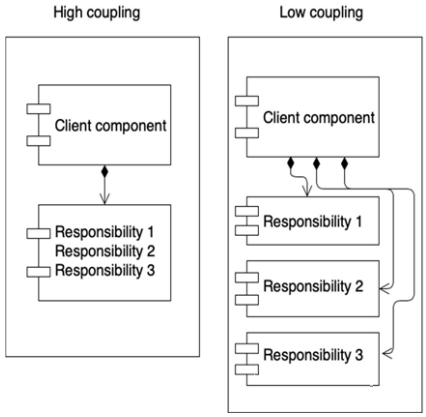


Fig. 4. Example of component decoupling

Message brokers are good candidates for bringing asynchronous processing into the application. Implementing a channel listener automatically makes the processing asynchronous, in other words, the message broker client has brought parallel processing. Moreover, the message broker-client decreases the overhead of implementing asynchronous processing through multiple instances with publish-subscribe patterns.

The advantage of paralleling the tasks per task type can be described with the Amdahl's law. Amdahl's law represents a relation between the speedup and a workload that can be anticipated from a system or component. The law is represented in the following formula:

$$S=\frac{1}{(1-p)+\frac{p}{s}} \quad (1)$$

where

- $S_{latency}$ is the theoretical speedup of the execution of the task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

The message brokers are beneficial in integrating new components into the existing system, especially in microservice architecture applications. Asynchronous response to existing system business processes can be implemented via changing new components in a way that new application components are subscribed to data manipulation operations from the core application to keep internal data consistent or to reflect outer changes in the internal application.

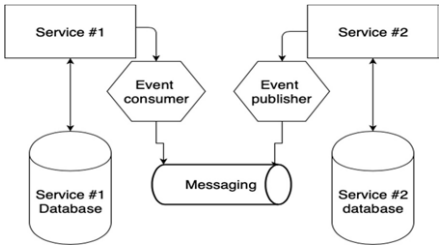


Fig. 5. Example of handling events

A more detailed example of integration new component with messaging lies in terms of the application system containing multiple instances which domain field is intersected among multiple services. In this case, data duplication becomes an inviable solution as inconsistency develops. Messaging can be applied to implement Read Model together with the event-driven approach. In this scenario, each microservice that has own domain logic should publish an event in case of successful data change to the dedicated channel. Thus, the number of events will come into existence which can be eventually be consumed by a single read model service. This service will be responsible for storing cross-data database records that can be queried by multiple services at a time, bringing eventual consistency into the system as asynchronous processing of each event needs the amount of time to be delivered to read model service.

Database data migration can be achieved in a live microservice production environment leveraging messaging. Each service that needs data to be migrated in addition to already existing one usually is capable of creating the same type of data. This can become a point of integration being adapted and injected into channel listener, that will deliver domain entities from the outdated data source to a new one. For this purpose, additional integration microservice should be deployed that will publish data on command.

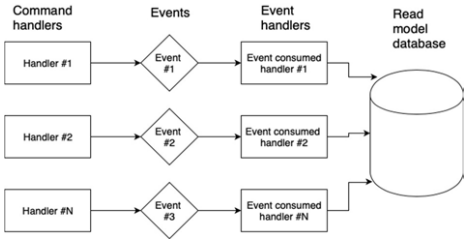


Fig. 6. Example of using CQRS pattern with underlying messaging

V. Results and Discussion

During the research, a single environment has been deployed with a microservice architecture approach application which resulted in the following findings.

Investigation the actual performance growth, consider RPC pattern implementation with the task X that comprises executing tasks A which lasts for 0.2 seconds, task B which lasts for 0.5 seconds and task C which lasts for 0.3 seconds. Assume that tasks B and C can be executed in parallel. In total, tasks B and C last for 0.8 seconds, hence they constitute 80% of task X. Hense, from the equation (1) p = 0.8. As tasks, B and C executing in parallel would end not earlier than the amount of the execution of the longest time-consuming task (here – within 0.5 seconds), the speedup of paralleling the tasks can be evaluated as 0.5 compared with 0.8 seconds of sequential executing. Hereof, the proportion of execution time that speedups can be calculated as $1 + (0.5 / 0.8) = 1.625$ which is s from the equation (1). Taking into account abovementioned, Amdahl's law states that the overall speedup of applying the improvement will be:

$$S=\frac{1}{(1-0.8)+\frac{0.8}{1.625}} \approx 1.44 \approx 44\% \quad (2)$$

Talking about the integration improvements, there are the following drawbacks of using message brokers: increased infrastructure support complexity and an additional point of failure in addition to the denormalized data usually residing in separate microservices.

As regards leveraging message brokers to applying the read model - the drawback of the stated approach is increased demand for the messaging event store as well as the need for additional monitoring for message broker availability. Transactional ACID properties could not be achieved as well (Atomicity, Consistency, Isolation, Durability) remaining the system consistency act within BASE consistency properties (Basically Available, Soft state, Eventual consistency).

VI. Conclusion and Future Work

As computer systems grow, the need in responsive, scalable and supportable application increased. The messaging approach has been developed to accommodate classic and modern software application patterns that are employed to maintain engineering maturity.

Messaging enables new architectural patterns that bring additional effort to implement and maintain. However, the message broker requires taking measures to guarantee high availability and broker instance monitoring.

In this article, we addressed the message broker design overview and hands-on examples of design solutions that can be integrated into an enterprise application. From the practical point of view, performance investigation between different messaging patterns should be considered when implementing one into the existing application as a program system varies with the application domain and functional requirements. Besides, testing of the system services which have underlying messaging usage is the point of the consideration to act within application non-functional requirements.

The results of the work can be used in the field of integrating heterogeneous systems; environments that require benefits of using parallel computing.

REFERENCES

[1] M. Fowler, *Patterns of Enterprise Application Architecture*, Addison Wesley, 2002, pp. 21-30.

[2] ISO/IEC TR 19759, Software Engineering — Guide to the Software Engineering Body of Knowledge (SWEBOK), 2005.

[3] J. Korab, *Understanding Message Brokers. Learn the Mechanics of Messaging though ActiveMQ and Kafka* , O'Reilly, 2017, pp.27-40.

[4] R. Monson-Haefel, M. Richards and D. A Chappell, *Java Message Service, 2nd Edition*, O'Reilly, 2009, pp. 130-134.

[5] J.-Young Byun, "A Real-time Message Delivery Method of Publish/Subscribe Model in Distributed Cloud Environment", M.S. thesis, Information Technology Research Center, Univ. Information & communications Technology Promotion, Daejeon, Republic of Korea, 2017.

[6] R. Rocha, "Improving the performance of a Publish-Subscribe message broker", in *IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, Valencia, Spain, May 2019.

ДОДАТОК Б

Слайди презентації

Презентація «Дослідження технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень»

Дослідження технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень

Керівник роботи:
проф. Дударь З.В.

Ст. гр. ПЗСм-18-1
Апухтін В. Г.

Харків
2019

Загальні відомості

- Розробка корпоративних додатків включає в себе значну кількість проблем.
- Концепція обміну повідомленнями вирішує проблем в цій галузі.

Постановка проблеми

До переліку питань розробки і супроводу розподілених корпоративних додатків входять:

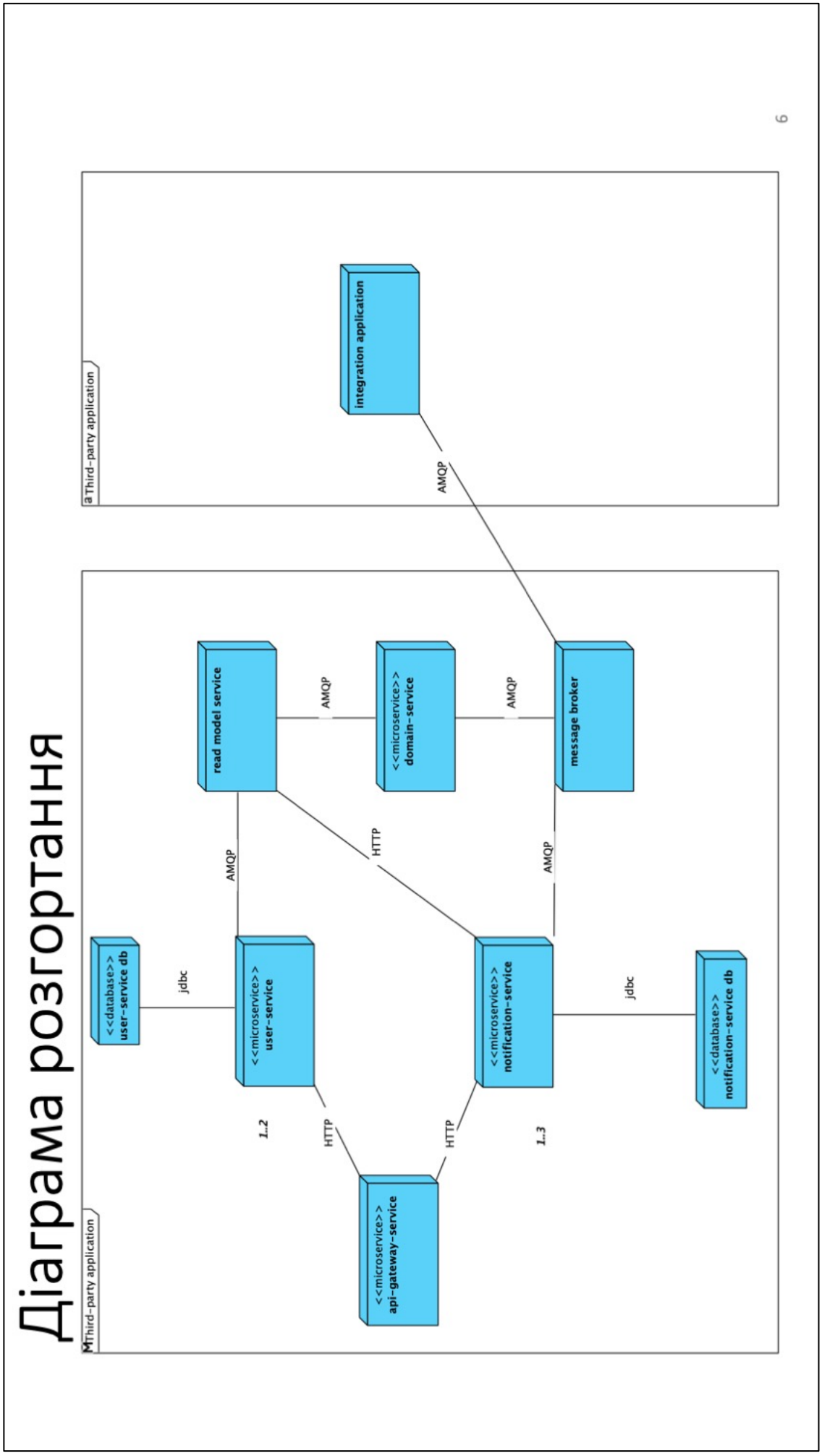
- Висока зв’язність компонентів
- Синхронна обробка запитів
- Проблема інтеграції додатків

Постановка задачі

- проаналізувати типові архітектурні рішення корпоративних застосувань;
- проаналізувати можливості впровадження брокерів повідомлень у корпоративні застосування;
- розробити архітектуру модельованої розподіленої системи **без** використання брокеру повідомлень;
- кількісно зробити заміри роботи системи за умови роботи користувача та бізнес процесу;
- математично передбачити приріст продуктивності системи та її компонентів;
- інтегрувати брокер повідомлень до розподіленої системи;
- кількісно зробити заміри роботи системи за умови роботи **після** інтеграції брокеру повідомлень;
- проаналізувати результати.

Деталі досліджуємої програмної системи

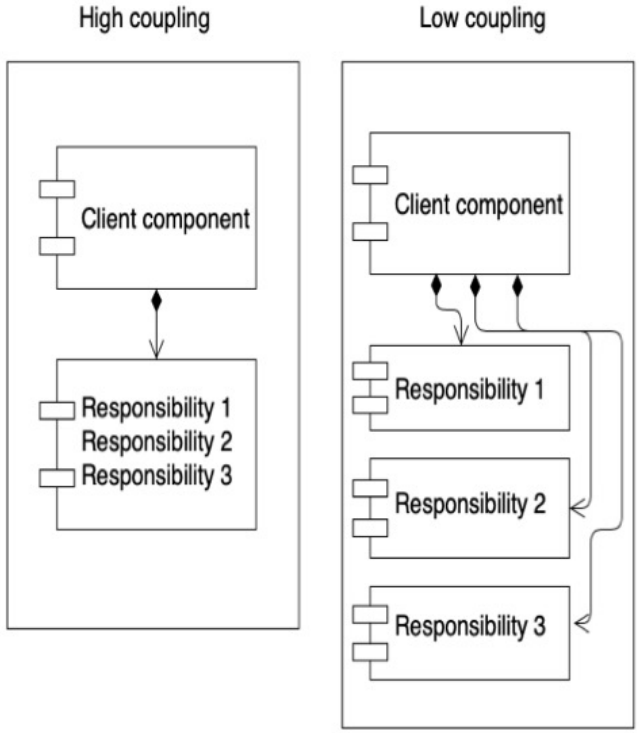
- Програмна система заснована на мікросервісній архітектурі
- Програмні інструменти: Java, Spring Cloud фреймворк
- Інфраструктурна складова: Docker, Docker compose
- Замір швидкості виконання запитів: VisualVM
- Емуляція інтеграційного сервісу: Python


Діаграма розгортання
Third-party application
<<database>>
user-service db
jdbc
<<microservice>>
user-service
1..2
HTTP
<<microservice>>
api-gateway-service
HTTP
<<microservice>>
notification-service
1..3
jdbc
<<database>>
notification-service db
AMQP
HTTP
AMQP
read model service
AMQP
<<microservice>>
domain-service
AMQP
message broker
Third-party application
integration application
AMQP
6

Методи

Обмін повідомленнями дозволяє розділити компоненти на окремі одиниці, що залежать від конкретного домену, які обмінюються даними через посередники по наступних шаблонах:

- Publisher-subscriber
- Exclusive point-to-point



Методи

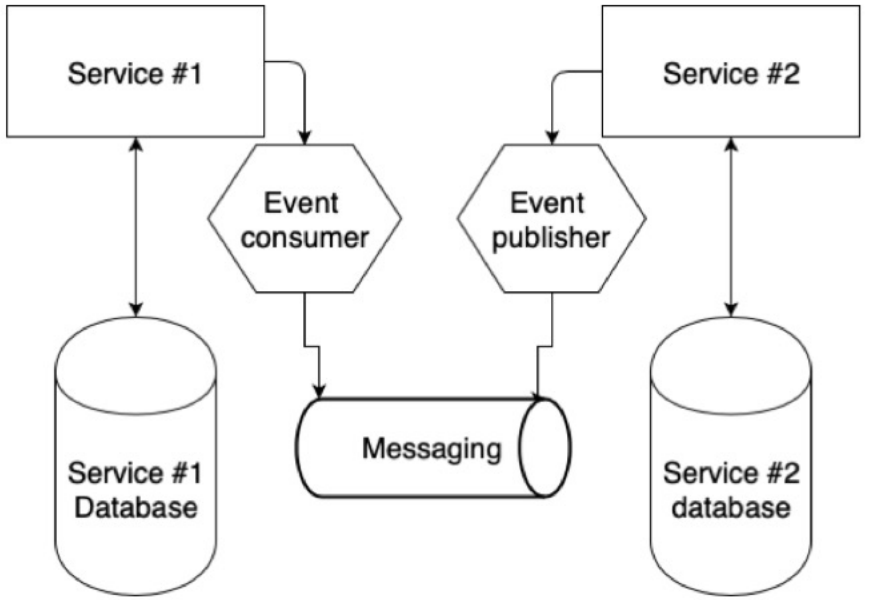
Інтеграція асинхронної обробки з використанням нових концепцій обмежених повідомлень:

- Реалізація прослуховувачів каналу автоматично виконує обробку асинхронно, тобто, брокер повідомлень забезпечує паралельну обробку.
- Брокер-кліент зменшує накладні витрати на реалізацію асинхронної обробки через кілька екземплярів

Методи

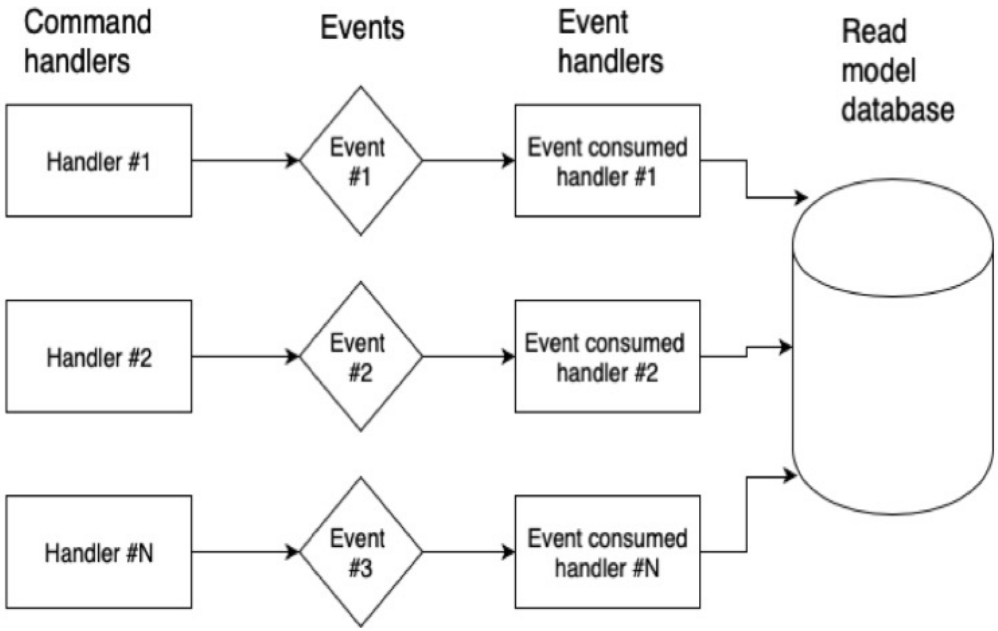
Інтеграція нових додатків може бути реалізована з використанням шаблонів брокера повідомлень:

- Publisher-subscriber
- Exclusive point-to-point
- RPC (Remote procedure call)



Методи

- Інтеграція додатків з різними доменами може бути реалізована за допомогою шаблону читацької моделі, де кожен тематичний канал обробляє поновлення даних з того чи іншого джерела даних.



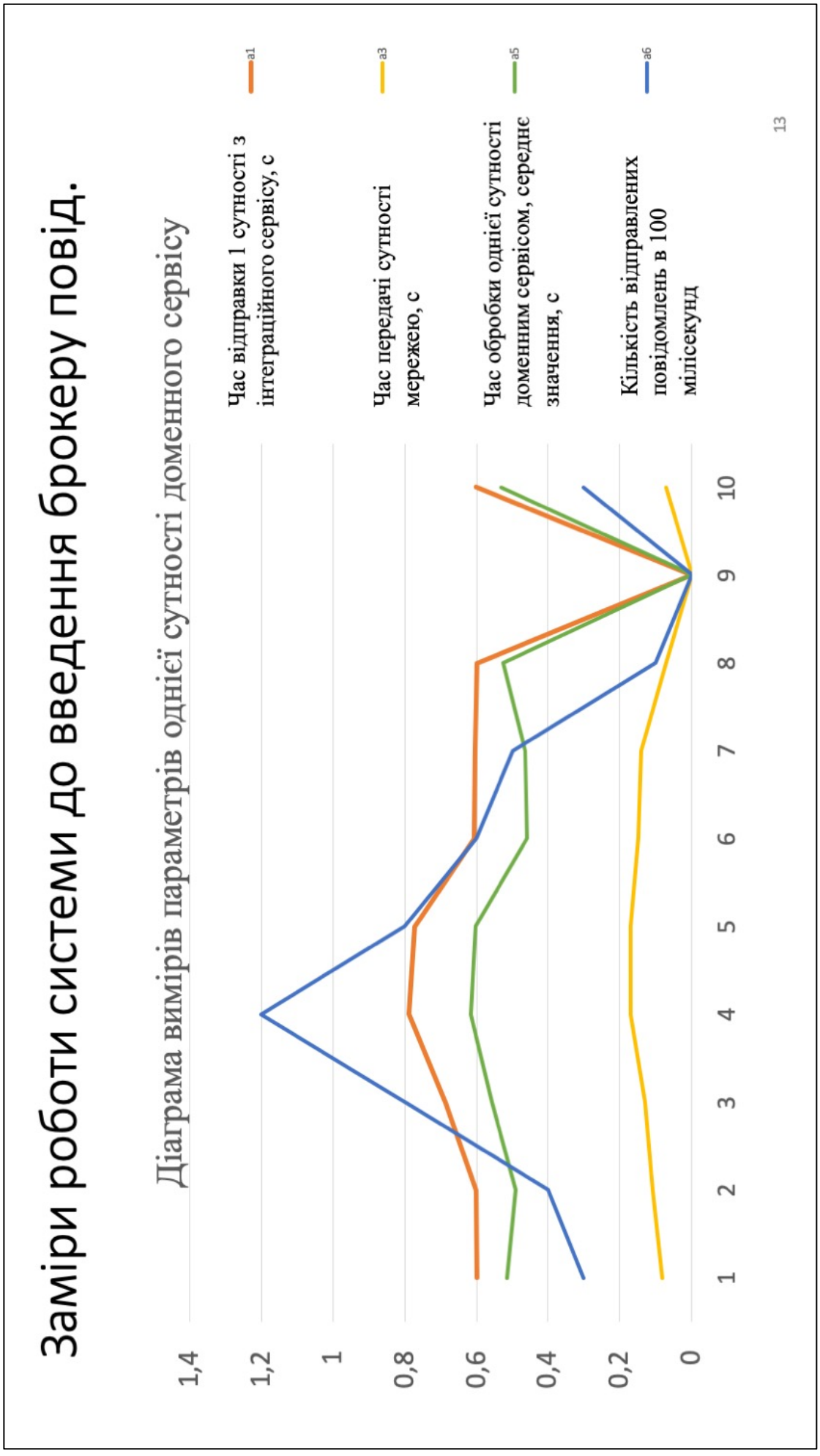
Результати

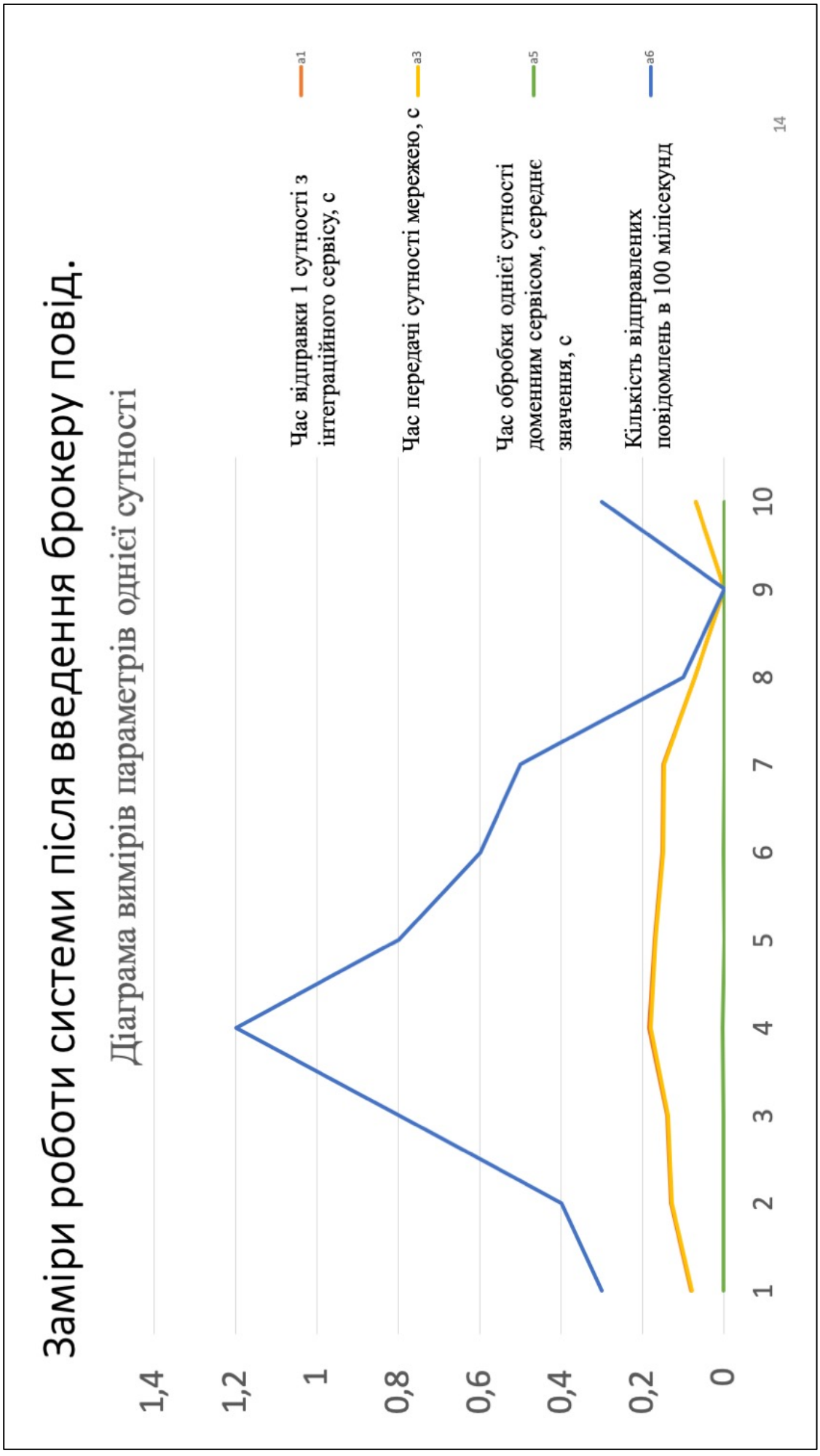
- Підвищення продуктивності асинхронної обробки було обчислено згідно із законом Амдаля. Прискорення було обчислено у 44%.
- $S = \frac{1}{(1-p)+\frac{p}{s}}$

$$S=\frac{1}{(1-0.8)+\frac{0.8}{1.625}} \approx 1.44 \approx 44\%$$

Результати

- Калькульяції були порівняні з реальними розрахунками підвищенням продуктивності, пов’язанним з асинхронною обробкою – 50,6% (наступні слайди)


Заміри роботи системи до введення брокеру повід.
Діаграма вимірів параметрів однієї сутності доменного сервісу
Час відправки 1 сутності з інтеграційного сервісу, с
Час передачі сутності мережею, с
Час обробки однієї сутності доменним сервісом, середнє значення, с
Кількість відправлених повідомлень в 100 мілісекунд
a1
a3
a5
a6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
1
2
3
4
5
6
7
8
9
10
13


Заміри роботи системи після введення брокеру повід.
Діаграма вимірів параметрів однієї сутності
1,4
1,2
1
0,8
0,6
0,4
0,2
0
1
2
3
4
5
6
7
8
9
10
14
Час відправки 1 сутності з інтеграційного сервісу, с
a1
Час передачі сутності мережею, с
a3
Час обробки однієї сутності доменним сервісом, середнє значення, с
a5
Кількість відправлених повідомлень в 100 мілісекунд
a6

Результати

- Підтримка розв'язаних компонентів істотно полегшена
- Архітектура Read Model, заснована на посередниках повідомлень, створює єдине джерело правди серед розподілених різнорідних додатків.

Аналіз

Переваги

- Асинхронна обробка
  - Відклик системи збільшується
- Зв’язність компонентів:
  - Використання посередників позбавляє залежності інші компоненти, що приводить до бесперебойного обслуговування та широкого розширення бази коду.
- Інтеграція додатків з моделлю чтенія
  - Єдиний джерело істинності вирішує проблему дублювання даних

Аналіз

Недоліки

- Асинхронна обробка
  - Збільшення витрат на розробку в синхронізації паралельних потоків виконання
- Зв’язність компонентів:
  - Вихід з ладу брокера повідомлень призводить до неможливості взаємодії компонентів
- Інтеграція програми з read model
  - Проблема ‘Eventual concistency’
  - ACID вимоги транзакцій не виконуються

17

Апробація результатів роботи

IEEE International Scientific and Practical Conference PIC S&T'2019

The Relevance of Using Message Brokers in Robust Enterprise Applications

Vladyslav Apukhtin, Mariya Shirokopetleva, Victoria Skovorodnikova
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
vladyslav.apukhtin@nure.ua, marija.shirokopetleva@nure.ua viktoriia.skovorodnikova@nure.ua

***Abstract*–The paper covers main message broker protocols and the patterns which are used to manage technical problems. The article provides the reader with typical software engineering design issues which might lead to conceptual violations, software non-functional requirements nonconformity, which is high coupling problem, synchronous operation performance loss, and system constituent integration problem. The solutions which come as a result of applying messaging are brought covering each problem above, in particular establishing the communication between loosely coupled applications, researching the application performance speedup with the Amdahl's law and applying Read Model architectural pattern leveraging message brokers capabilities.**

Keywords – messaging, asynchronous operations, Amdahl's law, protocol, application integration.

I. Introduction

Enterprise applications are applications that display, manipulate, and store large amounts of data and support or automate business processes with that data [1]. Such applications usually consist of separated modules, services or even clusters of services that handle business tasks differently. Taking into account the complexity of the architecture, there is a requirement to support integration between the components of the system. The idea of message brokers has come about as an adequate and flexible solution for this purpose.

The message brokers are a good instrument for resolving typical computer system problems. Section II describes the main messaging concepts and addresses the papers of the scientific field covered. Section III presents which issues can arise during the application development process. The practical solutions of them are presented in section IV which are aimed to solve distributed services communication ones in addition to centralizing distributed data via the messaging capabilities.

II. Literature Review

The Message broker is an intermediate software that is responsible for sending the messages of a format defined between software components using the specific protocol, commonly MQTT (Message Queuing Telemetry Transport), STOMP (Streaming Text Oriented Messaging Protocol), AMQP (Advanced Message Queuing Protocol).

Messaging systems typically involve the introduction of an intermediary between the two systems that communicate in order to further decouple the sender from the receiver or receivers. In doing so, the messaging system allows a sender to send a message without being aware of receiver, whether it is active, or indeed how many instances of them there are [3]. Different techniques (in other words – patterns) exist to achieve this goal, namely:

- Publisher-subscriber.
- Exclusive point-to-point.
- RPC (Remote procedure call).

The message is the amount of data that is considered to be exchanged. Usually, messages are restricted in size. Besides, there are different messaging publishing modes:

- *Exactly-once* i.e. the message can be delivered only once.
- *At-least-once* i.e. the message can be delivered multiple times with duplications.

Messages can be persistent on non-persistent which denotes that after message broker's breakdown there is the technical opportunity to redeliver messages or not respectively.

Message can contain properties and headers. Message properties are specific message characteristics, for instance, priority or content type.

Headers are message characteristics that can be flexibly changed. In opposite to message properties, headers can be taken into account by a broker to resolve the recipient.

Publisher-subscriber pattern brings the concepts of sender, receiver, message, and topic.

Publisher and Subscriber are the nodes of communication that exist in the many-to-many relationship; the nodes are responsible for sending and receiving messages respectively using exchange topics.

Exchange topics are string values that are used by the message broker to identify which subscriber is intended to receive a message that has been sent. The publisher sends a message to the broker with the exchange topic, the message broker resolves which consumers are capable of receiving the message.

Exchange topic key can be specified by consumer either as a direct value of the topic's name or following to the pattern.

It is assumed that a single message can be delivered to multiple receivers.

978-1-7281-4184-8/19/$31.00 ©2019 IEEE

Висновки

- У роботі було розглянуто дизайн системи з брокером повідомлень і практичні приклади проектних рішень, які можна інтегрувати в корпоративні додатки.
- Результати роботи можуть бути використані в області інтеграції гетерогенних систем; середовища, які вимагають переваг використання паралельних обчислень.

Запитання

ДОДАТОК В

Відгуки та рецензії

ВІДГУК

на атестаційну роботу магістра
студента групи ПЗСм-18-1Апухтіна Владислава Геннадійовича
спеціальність – 121- Інженерія програмного забезпечення
освітньо-професійна програма «Програмне забезпечення систем»
«Дослідження технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень».

Атестаційна робота магістра Апухтіна В.Г. присвячена дослідженню технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень. Наукова новизна отриманих результатів дослідження полягає в порівнянні швидкості відгуку системи, складності підтримки комп'ютерних систем та інших проблем розробки разом з використанням брокерів повідомлень. Результати дослідження можуть бути використані в проектування гетерогенних та розподілених програмних систем для випадків, описаних у роботі.

Робота була написана самостійно, опис дослідження був проведений детально, був продемонстрований високий рівень аналітичних здібностей.

Студент продемонстрував здатність до самостійних досліджень, здібність до використання теоретичних навичок у практиці.

План виконання дослідження був дотриманий.

Студент Апухтін В.Г. показав серйозне відношення до роботи, високий рівень підготовленості до самостійної діяльності. Результати роботи відповідають завданню на атестаційну роботу.

Магістрант гр. ПЗСм-18-1 Апухтін В.Г. готовий до самостійної інженерної діяльності. Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121-«Інженерія програмного забезпечення», освітньо-професійною програмою «Програмне забезпечення систем».

«_____» грудня 2019 р.

Керівник атестаційної роботи магістра
к.т.н., проф. Дударь З.В.

РЕЦЕНЗІЯ

на атестаційну роботу магістра
студента групи ПЗСм-18-1 Апухтіна Владислава Геннадійовича
спеціальність – 121- Інженерія програмного забезпечення
освітньо-професійна програма «Програмне забезпечення систем»
«Дослідження технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень».

Структура атестаційної роботи: пояснювальна записка 89 сторінок; графічна частина 18 аркушів; програмне застосування (прикладна програма) 478 файлів загальним обсягом 352 Мбайт.

Задачами роботи є дослідження методів інтеграції брокерів повідомлень у комп'ютерну систему та аналіз результатів.

Прийняття інженерних та наукових рішень роботи є доцільне.

В проведеному аналізі предметної області представлений докладний аналіз джерел мережі інтернет та зарубіжних публікацій, тематика яких відповідає досліджуваній темі.

В ході практичної реалізації дослідження були використані засоби: платформа Java, Spring Framework, Docker compose інфраструктура та VisualVM інструмент.

Надані результати роботи повністю відповідають завданню до атестаційної роботи.

Запропоновані методи є доцільними.

Пояснювальна записка відповідає стандартам та вимогам, які висуваються щодо атестаційних робіт магістра. У роботі автор продемонстрував вміння аналітично мислити, порівнювати переваги та недоліки існуючих шаблонів, методів.

Обсяг експериментальних досліджень є достатній. Якість отриманих результатів висока.

Слайди презентації відображають результати дослідження та опис реалізованих програм.

Як недоліки атестаційної роботи магістранта можна відзначити недостатньо детальний опис розрахунку пришвидшення системи завдяки впровадженню брокера повідомлень. Також, опис ReadMode шаблону є недостатньо детальний.

Атестаційна робота магістранта групи ПЗСм-18-1 Апухтіна В.Г. відповідає вимогам до атестаційних робіт і заслуговує оцінки «відмінно – 96, А». Атестаційну роботу можна представити для захисту в ЕК за спеціальністю 121- Інженерія програмного забезпечення, освітньо-професійною програмою «Програмне забезпечення систем».

Рецензент доц. каф. ПЗ Мазурова О.О.

РЕЦЕНЗІЯ

на атестаційну роботу магістра
студента групи ПЗСм-18-1 Апухтіна Владислава Геннадійовича
спеціальність – 121- Інженерія програмного забезпечення
освітньо-професійна програма «Програмне забезпечення систем»
«Дослідження технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень».

Структура атестаційної роботи: пояснювальна записка 89 сторінок; графічна частина 18 аркушів; програмне застосування (прикладна програма) 478 файлів загальним обсягом 352 Мбайт.

Задачами дослідження є порівняльний аналіз переваг та недоліків впровадження брокерів повідомлень у комп'ютерні системи.

Результати роботи повністю відповідають завданню до атестаційної роботи.

Пояснювальна записка відповідає стандартам та вимогам, які висуваються щодо атестаційних робіт магістра. Розділи пояснювальної записки пропорційні. У роботі автор продемонстрував вміння працювати з літературою, в достатній мірі процитував її в пояснювальній записці. Програмне забезпечення, надане на рецензування, відповідає завданню та є працездатним.

Було наведено детальний експериментальний опис дослідження вирішення однієї з описаних проблем, якість отриманих результатів є висока, деталізація графіків отриманих даних доцільна.

Автор успішно виділив актуальні проблеми розробки високонавантажених систем. Був проведений детальний експериментальне дослідження зміни часу відповіді між різними версіями системи. Слайди презентації відображають результати дослідження та опис реалізованих програм.

Робота має високий рівень можливості впровадження через високу практичну необхідність вирішення нагальних проблем корпоративних застосувань.

Серед недоліків можна виділити відсутність глибокої деталізації реалізації програмної системи.

Атестаційна робота магістранта групи ПЗСм-18-1 Апухтіна В.Г. відповідає вимогам до атестаційних робіт і заслуговує оцінки «відмінно – 94, B». Атестаційну роботу можна представити для захисту в ЕК за спеціальністю 121- Інженерія програмного забезпечення, освітньо-професійною програмою «Програмне забезпечення систем».

Рецензент проф. каф. ІУС, д.т.н.
Єванов М.В.

ВІДГУК

на атестаційну роботу магістра
студента групи ПЗСм-18-1Апухтіна Владислава Геннадійовича
спеціальність – 121- Інженерія програмного забезпечення
освітньо-професійна програма «Програмне забезпечення систем»
«Дослідження технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень».

Атестаційна робота магістра Апухтіна В.Г. присвячена дослідженню технологій та методів обміну даних в корпоративних системах за допомогою брокерів повідомлень. Наукова новизна отриманих результатів дослідження полягає в порівнянні швидкості відгуку системи, складності підтримки комп'ютерних систем та інших проблем розробки разом з використанням брокерів повідомлень. Результати дослідження можуть бути використані в проектування гетерогенних та розподілених програмних систем для випадків, описаних у роботі.

Робота була написана самостійно, опис дослідження був проведений детально, був продемонстрований високий рівень аналітичних здібностей.

Студент продемонстрував здатність до самостійних досліджень, здібність до використання теоретичних навичок у практиці.

План виконання дослідження був дотриманий.

Студент Апухтін В.Г. показав серйозне відношення до роботи, високий рівень підготовленості до самостійної діяльності. Результати роботи відповідають завданню на атестаційну роботу.

Магістрант гр. ПЗСм-18-1 Апухтін В.Г. готовий до самостійної інженерної діяльності. Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121-«Інженерія програмного забезпечення», освітньо-професійною програмою «Програмне забезпечення систем».

«9» грудня 2019 р.

Керівник атестаційної роботи магістра
к.т.н., проф. Дударь З.В.