УДК 681.51
DOI https://doi.org/10.32782/2663-5941/2023.3.1/33

Пашковський Б.В.
Івано-Франківський національний технічний університет нафти і газу

Слабінога М.О.
Івано-Франківський національний технічний університет нафти і газу

РОЗРОБКА МІКРО СЕРВІСНОЇ АРХІТЕКТУРИ ІНТЕГРАТОРА ОПТИМІЗАЦІЙНОЇ ЗАДАЧІ ПРОЦЕСУ КОМПРИМУВАННЯ ПРИРОДНЬОГО ГАЗУ В УМОВАХ НЕВИЗНАЧЕНОСТІ

У статті розглянуто процес розробки програмного забезпечення інтегратора оптимізаційної задачі процесу компримування природного газу в умовах невизначеності.

Основу програмного забезпечення інтегратора оптимізаційної задачі комп'ютерної системи процесу компримування природного газу в умовах невизначеності складає Restful сервіс виконаний за допомогою ASP .NET Web API 2 і об'єктно-орієнтованої мови C#.

Сервіс інтегратора, разом із клієнтським веб-додатком забезпечують реалізацію таких функцій: збір та обробку даних від давачів технологічних параметрів та від вібродавачів, імпорт даних за минулі періоди, для більш точної побудови моделі оптимальної складності, відображення значень контрольованих параметрів, візуалізація в реальному часі інформації про стан газоперекачувальних агрегатів та їхніх компонентів, відображення рекомендованих частот обертання відцентрових нагнітачів кожного з ГПА.

Доцільним є розділення монолітного застосунку на мікро сервіси, кожен з яких виконував би ізольовану задачу.

Запропонована мікро сервісна архітектура програмного забезпечення інтегратора оптимізаційної задачі комп'ютерної системи процесу компримування природного газу в умовах невизначеності, що дозволить отримати наступні переваги:

– гнучкість та масштабованість: мікросервіси дозволяють гнучко масштабувати окремі компоненти системи незалежно один від одного. Це означає, що ми можемо збільшувати ресурси тільки для тих мікросервісів, які потребують додаткового обсягу роботи, замість масштабування всієї програми.

– легка розгортка та незалежність: кожен мікросервіс може бути розгорнутий, масштабований та оновлюваний незалежно. Це полегшує розробку, тестування та впровадження нових функцій, оскільки ми можемо працювати з кожним мікросервісом окремо без впливу на решту системи.

– технологічна різноманітність: мікросервісна архітектура дає змогу використовувати різні технології та мови програмування для різних мікросервісів. Це дає можливість використовувати найкращі інструменти для кожної конкретної задачі, замість обмеження однією технологією для всього проекту.

Ключові слова: *компримування природного газу, газоперекачувальний агрегат, оптимальне керування, мікросервіси, інфраструктура.*

Постановка проблеми. В умовах повномасштабного вторгнення Російської Федерації на територію України та зміни структури ринку енергоносіїв в Європі та Україні зокрема, актуальною науково-технічною задачею є оптимізація витрат паливного газу в умовах невизначеності і з врахуванням технічного стану газоперекачувальних агрегатів (ГПА).

У роботі [1] запропонована система оптимізації процесу компримування природного газу та здійснено її технічну реалізацію за допомогою системи приладів, інтерфейсу введення-виведення та персональної ЕОМ, яка дає можливість знаходити оптимальні значення навантаження нагнітачів на основі їхнього реального технічного стану.

Синтезована структура комп'ютерної системи, що забезпечує поетапний розв'язок задачі оптимізації процесу компримування, та інтегрована в існуючу систему керування процесом.

Розроблений прикладний програмний продукт підтримки задачі оптимізації процесу, що забезпечує розв'язання задач оптимального керування за єдиним сценарієм з можливістю інтегрування його в існуюче програмне середовище.

Проте існуюча монолітна реалізація продукту має ряд недоліків:

– складність масштабування: в монолітній архітектурі всі компоненти програмної системи зв'язані між собою прямими залежностями.

Це робить процес масштабування важким, оскільки потрібно масштабувати всю програму, навіть якщо це необхідно лише для окремої частини.

– обмежена технологічна різноманітність: у монолітній архітектурі зазвичай використовується одна мова програмування та одна технологія. Це може бути обмеженням, оскільки ми не можемо використовувати найкращі інструменти для кожної конкретної задачі.

– висока залежність та ризик впливу помилок: У монолітній архітектурі відмова або помилка в одному компоненті може мати вплив на всю систему. Це може бути проблематичним, оскільки важко локалізувати та ізолювати проблему, і це може спричинити збої в роботі всього додатку.

– складність розробки та підтримки: Великий монолітний додаток може бути складним у розробці та підтримці. Збільшення розміру та складності кодової бази може зробити його важким для розуміння та модифікації. Крім того, великі команди розробників можуть зіткнутися з проблемою конфліктів під час одночасної роботи над одним кодом.

Таким чином актуальною є задача розділення монолітної програмної архітектури на мікросервіси.

Аналіз останніх досліджень і публікацій. У роботі [2] проведене емпіричне дослідження, що дало систематичне розуміння промислових практик щодо мікросервісів.

Досліджено відмінності між ідеальним баченням і реальними промисловими практиками щодо мікросервісів і того, які переваги можна отримати від промислового досвіду. Проведено серію промислових інтерв'ю з тринадцятьма різними типами компаній. Зібрані дані було кодифіковані відповідно до визначених якісних методів. В результаті охарактеризовано розбіжності між типовими характеристиками, прийнятими в індустрії, та промисловою практикою мікросервісів.

Автори роботи [3] аналізували збір даних із розумних пристроїв інтернету речей (IoT). Важливість цього збору даних залежить від того факту, що він може виявити цінну інформацію та сприяти розумнішому та швидшому прийняттю рішень. Це дозволяє організаціям швидко адаптуватися до змін у робочих процесах, скоротити час простою, розширити виробничі потужності та підвищити загальну ефективність роботи. Однак проблема полягає в тому, що на багато з цих додатків промислового інтернету речей (IoT) може значно впливати склад API RESTFul та архітектура мікросервісів, яку вони інтегрують. Крім того, додатки IoT не враховують динамізм сервісних середовищ і вимог.

У роботі [4] проведено огляд міграції програм на архітектуру Microservices багатьма компаніям, для того щоб залишатися конкурентоспроможними в середовищі, що швидко змінюється. Такі масштабні міграційні процеси вимагають ретельного планування та розгляду як наслідків, так і викликів. У цьому відношенні практичний досвід із промислової практики все ще рідкість. Щоб заповнити цю прогалину в науковій літературі, авторами було проведене якісне дослідження намірів, стратегій і викликів у контексті переходу на мікросервіси. Досліджено процес міграції 14 систем у різні домени та розміри, проведено 16 детальних інтерв'ю з фахівцями з програмного забезпечення з 10 компаній.

Метою роботи є розроблення мікросервісної архітектури програмного забезпечення інтегратора оптимізаційної задачі комп'ютерної системи процесу компримування природного газу в умовах невизначеності.

Поставлена мета досягнута внаслідок розв'язання таких задач:

– здійснено аналіз літературних джерел і виявлено, що мікросервісна архітектура має ряд переваг;

– здійснено розбиття предметної області на мікросервіси;

– оцінені переваги нової мікросервісної архітектури над існуючою монолітною.

Інтегратор оптимізаційної задачі.

Інформація, яка отримана від давачів технологічних параметрів та від вібродавачів, відображається засобами візуалізації на автоматизованому робочому місці оператора за допомогою базового програмного забезпечення RealFlex та обмінюється з прикладними програмами, що розроблені для компресорної станції та окремих агрегатів.

На рівні диспетчера ЛВУ, крім штатного, розроблене додаткове програмне забезпечення задачі оптимізації процесу компримування природного газу.

Інтегратор оптимізаційної задачі написаний у вигляді Restful сервісу за допомогою ASP .NET Web API 2 і об'єктно орієнтованої мови C#.

Перевагою такої реалізації є те, що сервіс є незалежним і може використовуватися будь-яким клієнтом за допомогою протоколу HTTP.

Клієнтом може виступати і програма на Windows і Веб-додаток, і будь-який мобільний додаток на мобільних операційних системах IOS і Android.

Сервіс інтегратора, разом із клієнтським вебдодатком забезпечують реалізацію таких функцій:

– збір та обробку даних від давачів технологічних параметрів та від вібродавачів;

– імпорт даних за минулі періоди, для більш точної побудови моделі оптимальної складності;

– відображення значень контрольованих параметрів;

– візуалізація в реальному часі інформації про стан газоперекачувальних агрегатів та їх компонентів;

– відображення рекомендованих частот обертання відцентрових нагнітачів кожного з ГПА.

Інтегратор дозволяє імпортувати дані з файлів типу *.xls, *.xlsx, *.csv. Залежно від типу файлу за допомогою шаблону програмування «Фабрика» визначається конкретний клас, який оброблятиме цей файл.

Дані отримані з давачів, та попередньо імпортовані зберігаються у базі даних (БД) типу Microsoft SQL. Доступ до бази даних реалізовано за допомогою ORM від компанії Microsoft – EntityFramework 6.0.

Оскільки кількість імпортованих даних може бути доволі велике, задля зменшення часу виконання програми і економії процесорного часу, проведена оптимізації програми і використано методи пакетної роботи з базою даних “Bulk insert” і “Bulk Update”

На першому кроці оператор має можливість імпортувати дані у систему, для більш точної побудови моделі (рис. 1).

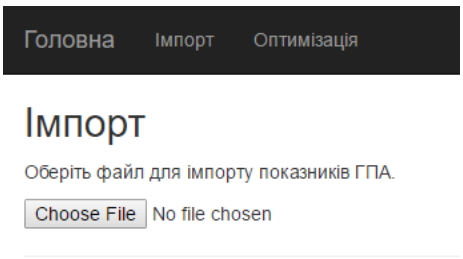

Рис. 1. Імпорт даних у систему

Далі оператор вводить кількість наявних нагнітачів, їхню продуктивність, затрати на привід та бажаний обсяг перекачки газу (рисунок 2) і система розраховує кількість працюючих нагнітачів у кожній групі, яких достатньо для підтримування заданого обсягу перекачування газу (рисунок 3).

Розділення монолітної архітектури на мікросервіси може бути виконано кількома способами (рисунок 4).

Функціональне розбиття: аналізується функціональність програмної системи та здійснюється ідентифікація незалежних компонентів або областей відповідальності. Кожен з цих компонентів може стати окремим мікросервісом. Наприклад, у електронному магазині окремими мікросервісами можуть бути обробка замовлень, управління користувачами, оплата тощо.

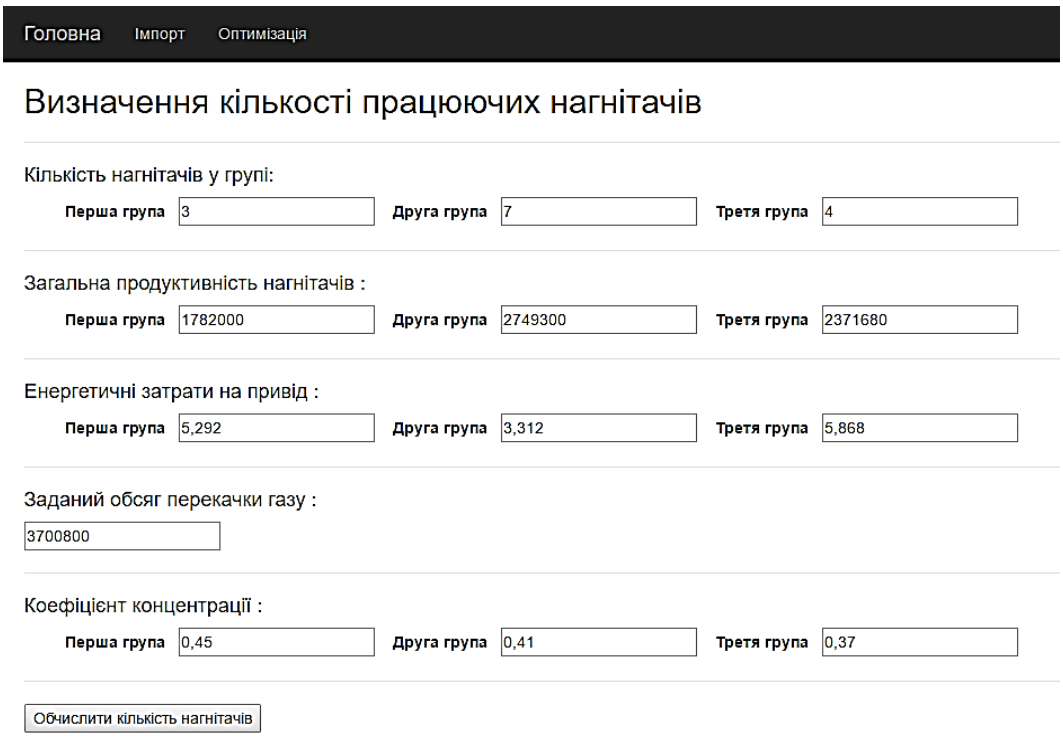

Рис. 2. Дані для розрахунку кількості нагнітачів

Доменно-орієнтоване розбиття: Засновуючись на принципах доменно-орієнтованого проектування (Domain-Driven Design), виділяються мікросервіси на основі окремих доменів або областей бізнесу. Кожен мікросервіс відповідає за певну сферу бізнес-логіки. Це дозволяє зосередитися на контексті домену та підтримувати межі між мікросервісами.

Розбиття на основі даних: Іноді розбиття моноліту може відбуватися на основі виділення окремих сутностей або компонентів бази даних в окремі мікросервіси. Кожен мікросервіс може мати власну базу даних або набір послуг для керування цими даними.

Декомпозиція за допомогою API: моноліт розглядається, як набір функцій або модулів, які можуть бути доступні через API. Поступово ці функції можна виділити в окремі мікросервіси, використовуючи API як інтерфейс взаємодії між ними.

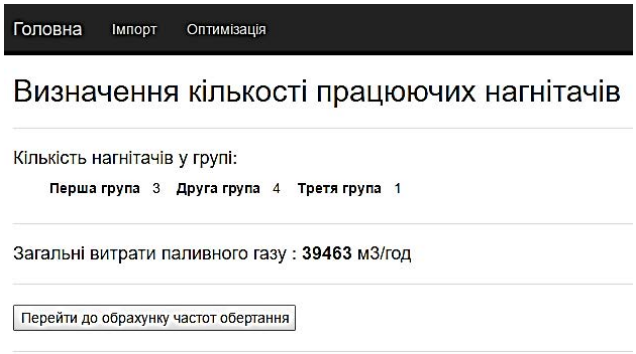

Рис. 3. Результат розрахунку кількості нагнітачів

В даному випадку використаємо функціональне розбиття.

Таким чином отримаємо перелік наступних мікросервісів.

– мікросервіс збір та обробку даних від давачів технологічних параметрів та від вібродавачів;

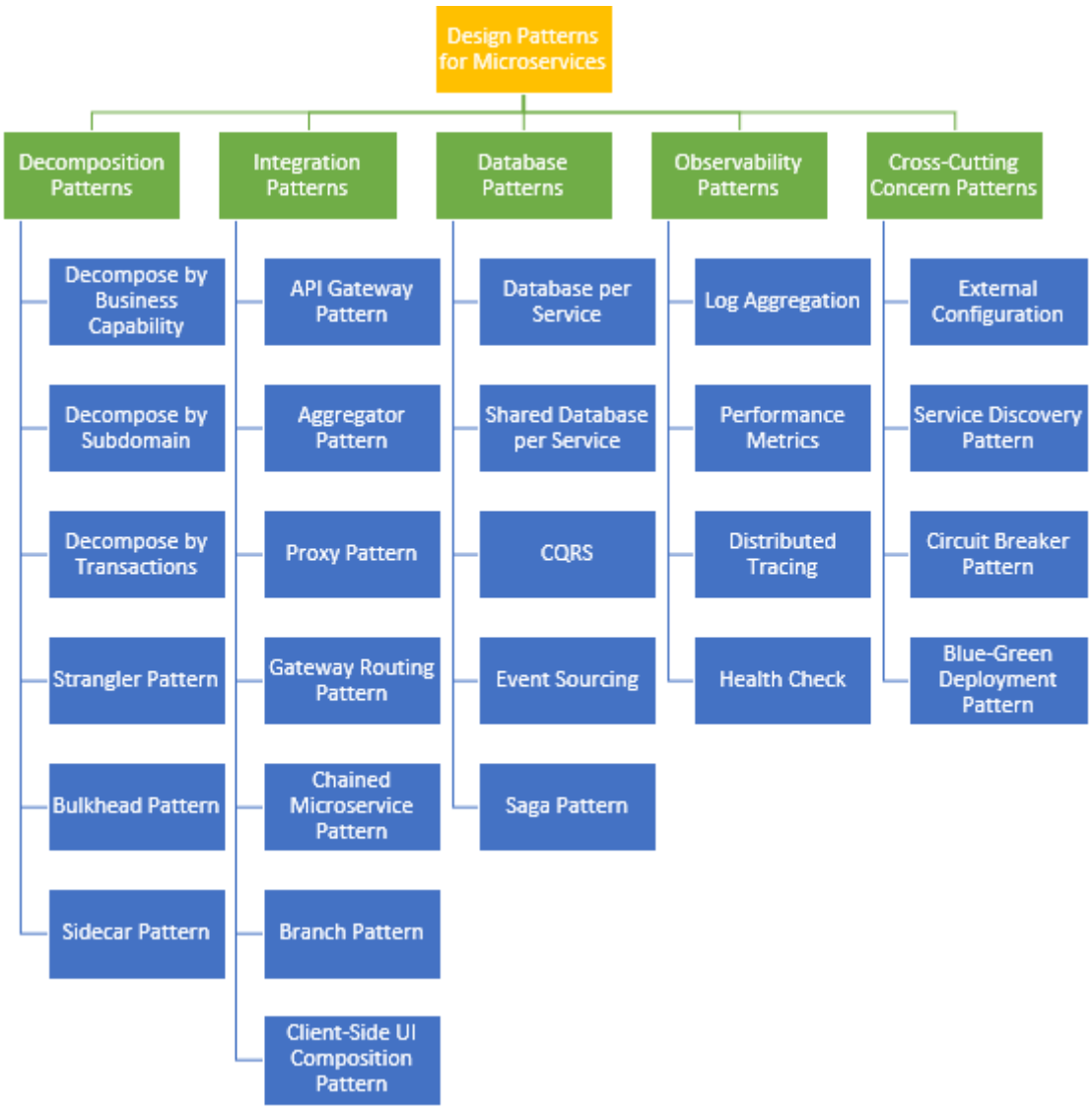

Рис. 4. Мікросервісна архітектура

– мікросервіс імпорту даних за минулі періоди, для більш точної побудови моделі оптимальної складності;

– мікросервіс користувацького інтерфейсу;

– мікросервіс візуалізації даних в реальному часі.

Для кожного мікросервісу було налаштовано окреме розгортання у хмарному сервісі Microsoft Azure, з використання технології CI/CD та продукту Azure Devops. Переваги CI/CD включають:

– швидше внесення змін: Автоматизований процес CI/CD дозволяє розробникам швидко вносити та тестувати зміни в коді, що сприяє швидкому розгортанню нових функцій та виправленню помилок.

– зменшення ризику: Частіша інтеграція та розгортання дозволяють виявляти та виправляти проблеми швидше, що зменшує ризик накопичення помилок у коді та сприяє стабільній роботі програмного забезпечення.

– автоматизоване тестування: CI/CD сприяє автоматизації процесу тестування.

Висновки. Проаналізовано існуючі способи розділення монолітної архітектури на мікросервісну. Проведено розділення архітектури інтегратора оптимізаційної задачі шляхом функціонального розбиття. Реалізоване незалежне розгортання мікросервісів у хмарному сервісі Microsoft Azure за допомогою підходів CI/CD та показано переваги такого підходу.

Список літератури:

1. Пашковський Б. В. Оптимальне керування процесом компримування природного газу в умовах невизначеності: дис. канд. техн. наук: 05.13.07. Івано-Франківськ, 2018

2. H. Zhang, S. Li, Z. Jia, C. Zhong and C. Zhang, "Microservice Architecture in Reality: An Industrial Inquiry", 2019 IEEE International Conference on Software Architecture (ICSA), Hamburg, Germany, 2019, pp. 51-60, doi: 10.1109/ICSA.2019.00014.

3. E. Al-Masri, "Enhancing the Microservices Architecture for the Internet of Things", 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 5119-5125, doi: 10.1109/BigData.2018.8622557.

4. J. Fritzsch, J. Bogner, S. Wagner and A. Zimmermann, "Microservices Migration in Industry: Intentions, Strategies, and Challenges", 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 481-490, doi: 10.1109/ICSME.2019.00081.

Pashkovskyi B.V., Slabinoha M.O. MICRO SERVICE ARCHITECTURE IMPLEMENTRATION OF THE INTEGRATOR OF THE OPTIMIZATION OF THE NATURAL GAS COMPRESSION PROCESS UNDER FUZZY CONDITIONS

The core of the integrator software for the optimization task of the computer system of the process of natural gas compression under fuzzy conditions is the Restful service, implemented using ASP .NET Web API 2 and the object-oriented language C#.

The integrator service, together with the client web application, provides the implementation of the following functions: collection and processing of data from sensors of technological parameters and from vibration sensors, import of data for past periods, for more accurate construction of a model of optimal complexity, display of values of controlled parameters, visualization of information in real time about the condition of gas pumping units and their components, displaying the recommended rotation frequencies of the centrifugal superchargers of each of the gas pumps.

It is recommended to divide a monolithic application into micro services, each of which would perform an isolated task.

The proposed micro service architecture of the software integrator of the optimization problem of the computer system of the natural gas compression process under fuzzy conditions, which will allow obtaining the following advantages:

– flexibility and scalability: microservices allow flexible scaling of individual system components independently of each other. This means that we can increase resources only for those microservices that need additional work, instead of scaling the entire application.

– easy deployment and independence: each microservice can be deployed, scaled and updated independently. This makes it easier to develop, test, and implement new features because we can work with each microservice separately without affecting the rest of the system.

– technological diversity: the microservice architecture enables the use of different technologies and programming languages for different microservices. This makes it possible to use the best tools for each specific task, instead of limiting yourself to one technology for the entire project.

Key words: *natural gas compression, gas pumping unit, optimal control, microservices, infrastructure.*