Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

Praca dyplomowa magisterska

na kierunku Informatyka Stosowana
w specjalności Inżyniera Oprogramowania

Analiza danych w czasie rzeczywistym z systemów wieloczujnikowych z wykorzystaniem klastra Kubernetes oraz narzędzi Big Data

Damian Wojcik

promotor
dr. inż. Łukasz Oskwarek

Warszawa, 2025

Spis treści

Streszczenie

Streszczenie

Temat: Analiza danych w czasie rzeczywistym z systemów wieloczujnikowych z wykorzystaniem klastra Kubernetes oraz narzędzi Big Data

Niniejsza praca magisterska przedstawia projekt, implementację i analizę wydajności systemu do przetwarzania danych w czasie rzeczywistym z wielu czujników, wykorzystując architekturę mikroserwisową opartą na klastrze Kubernetes oraz narzędzia Big Data, w szczególności Apache Kafka i Kafka Streams.

Praca koncentruje się na efektywnym pozyskiwaniu, przetwarzaniu i analizie strumieni danych generowanych przez systemy wieloczujnikowe w środowiskach przemysłowych, ze szczególnym uwzględnieniem procesów produkcyjnych, takich jak synteza amoniaku (proces Habera). Zaproponowany system umożliwia monitorowanie stanu instalacji przemysłowych w czasie rzeczywistym, wykrywanie anomalii oraz przewidywanie potencjalnych awarii.

Przeprowadzone badania eksperymentalne wykazują, że zastosowanie klastra Kubernetes jako platformy do wdrożenia systemu analitycznego zapewnia elastyczne skalowanie zasobów w zależności od obciążenia, a wykorzystanie narzędzi do przetwarzania strumieniowego, takich jak Apache Kafka, umożliwia efektywną analizę dużych wolumenów danych w czasie rzeczywistym.

Praca demonstruje praktyczne zastosowania opracowanego rozwiązania w przemyśle, przedstawiając korzyści ekonomiczne i operacyjne wynikające z wdrożenia systemu analitycznego opartego na klastrze Kubernetes oraz narzędziach Big Data.

Słowa kluczowe: Kubernetes, analiza danych w czasie rzeczywistym, Apache Kafka, systemy wieloczujnikowe, architektura mikroserwisowa, przetwarzanie strumieniowe, Big Data

Abstract

Streszczenie

Title: Real-Time Data Analysis from Multi-Sensor Systems Using Kubernetes Cluster and Big Data Tools

This master's thesis presents the design, implementation, and performance analysis of a real-time data processing system for multi-sensor environments, utilizing a microservice architecture based on Kubernetes cluster and Big Data tools, particularly Apache Kafka and Kafka Streams.

The thesis focuses on efficiently acquiring, processing, and analyzing data streams generated by multi-sensor systems in industrial environments, with particular emphasis on production processes such as ammonia synthesis (Haber process). The proposed system enables real-time monitoring of industrial installations, anomaly detection, and prediction of potential failures.

The experimental research demonstrates that using a Kubernetes cluster as a platform for deploying an analytical system provides flexible resource scaling depending on workload, while utilizing stream processing tools such as Apache Kafka enables efficient analysis of large volumes of data in real-time.

The thesis demonstrates practical applications of the developed solution in industry, presenting economic and operational benefits resulting from the implementation of an analytical system based on a Kubernetes cluster and Big Data tools.

Słowa kluczowe: Kubernetes, real-time data analysis, Apache Kafka, multi-sensor systems, microservice architecture, stream processing, Big Data

1 Wprowadzenie

1.1 Kontekst i motywacja badań

W dobie czwartej rewolucji przemysłowej (Przemysł 4.0) i Internetu Rzeczy (IoT), systemy wieloczujnikowe stają się nieodłącznym elementem nowoczesnych procesów produkcyjnych. Generują one ogromne ilości danych, które odpowiednio wykorzystane mogą dostarczyć cennych informacji na temat stanu i wydajności monitorowanych procesów. Analiza tych danych w czasie rzeczywistym stanowi jednak wyzwanie, zarówno pod względem technicznym, jak i organizacyjnym.

Tradycyjne podejścia do przetwarzania danych, oparte na scentralizowanych systemach, często nie są w stanie efektywnie obsłużyć dużej liczby równoczesnych strumieni danych przy zachowaniu niskich opóźnień. Ponadto, skalowanie takich systemów w odpowiedzi na rosnące obciążenie może być problematyczne i kosztowne.

Rozproszona architektura, w połączeniu z technologiami konteneryzacji i orkiestracji, takimi jak Kubernetes, oferuje nowe możliwości w zakresie budowy systemów do analizy danych w czasie rzeczywistym. Umożliwia ona elastyczne skalowanie, izolację usług i łatwiejsze zarządzanie złożonymi aplikacjami. Jednocześnie narzędzia Big Data, takie jak Apache Kafka czy Apache Spark zapewniają wydajne mechanizmy do przetwarzania strumieniowego, niezbędne w analizie danych w czasie rzeczywistym.

Motywacją do podjęcia niniejszych badań jest potrzeba sprawdzenia możliwości do opracowania wydajnego i skalowalnego do analizy danych w czasie rzeczywistym z systemów wieloczujnikowych, który mógłby być wykorzystany w różnych procesach przemysłowych.

1.2 Cel i zakres pracy

Głównym celami niniejszej pracy są zaprojektowanie, implementacja i analiza wydajności systemu do przetwarzania danych w czasie rzeczywistym z wielu czujników.

Szczegółowe cele pracy obejmują:

- Opracowanie architektury systemu do analizy danych w czasie rzeczywistym z systemów wieloczujnikowych, opartego na klastrze Kubernetes i narzędziach Big Data
- Implementacja systemu zgodnie z zaprojektowaną architekturą, z wykorzystaniem mikroserwisów i technologii przetwarzania strumieniowego
- Opracowanie i implementacja algorytmów do analizy danych w czasie rzeczywistym, w tym algorytmów do detekcji anomalii i przewidywania awarii
- Przeprowadzenie badań eksperymentalnych w celu oceny wydajności i skalowalności opracowanego systemu
- Analiza przydatności systemu w środowisku przemysłowym, ze szczególnym uwzględnieniem procesu syntezy amoniaku

Zakres pracy obejmuje:

- Analizę istniejących rozwiązań do przetwarzania danych w czasie rzeczywistym
- Projekt i implementację systemu przetwarzania danych opartego na klastrze Kubernetes
- Opracowanie mechanizmów agregacji i analizy danych z wielu czujników

- Implementację wizualizacji danych w czasie rzeczywistym
- Testowanie wydajności i skalowalności systemu
- Ocenę przydatności zaimplementowanego rozwiązania w środowisku produkcyjnym

1.3 Metodologia badań

Badania przeprowadzone w ramach niniejszej pracy opierają się na następującej metodologii:

1. Pomiar obciążenia procesora i zajętości pamięci operacyjnej
2. Testowanie poprawności wykonywania algorytmów pod zwiększonym obciążeniem
3. Testowanie automatycznego skalowania systemu
4. Pomiar przepustowości i opóźnień przetwarzania

W badaniach wykorzystano dane symulowane, generowane przez funkcje AWS Lambda.

1.4 Struktura pracy

Praca składa się z następujących rozdziałów:

- **Rozdział 1: Wprowadzenie** - przedstawia kontekst i motywację badań, cel i zakres pracy, tezę oraz metodologię badań
- **Rozdział 2: Przegląd literatury i stan wiedzy** - omawia istniejące rozwiązania w zakresie analizy danych w czasie rzeczywistym, architektury mikroserwisowej, Kubernetes oraz narzędzi Big Data
- **Rozdział 3: Projekt systemu przetwarzania danych w czasie rzeczywistym** - przedstawia wymagania, architekturę, model danych oraz przepływ danych w projektowanym systemie
- **Rozdział 4: Implementacja systemu na klastrze Kubernetes** - opisuje konfigurację klastra, wdrożenie mikroserwisów, konfigurację Apache Kafka i Kafka Streams, implementację przetwarzania strumieniowego
- **Rozdział 5: Algorytmy analizy danych w czasie rzeczywistym** - omawia zaimplementowane algorytmy do analizy danych
- **Rozdział 6: Badania eksperymentalne i analiza wyników** - przedstawia metodologię testowania, scenariusze testowe, wyniki badań wydajnościowych
- **Rozdział 7: Zastosowania praktyczne** - omawia praktyczne zastosowania opracowanego systemu w przemyśle, korzyści ekonomiczne i operacyjne oraz wyzwania wdrożeniowe
- **Rozdział 8: Podsumowanie i wnioski** - zawiera podsumowanie pracy, główne osiągnięcia i wnioski, ograniczenia badań oraz kierunki przyszłych badań

2 Przegląd litartury i stan wiedzy

W niniejszym rozdziale przedstawiono aktualny stan wiedzy w zakresie analizy danych w czasie rzeczywistym, systemów wieloczujnikowych, architektury mikroserwisowej, platformy Kubernetes oraz narzędzi Big Data. Omówiono również istniejące rozwiązania i ich ograniczenia.

2.1 Analiza danych w czasie rzeczywistym

Analiza danych w czasie rzeczywistym to proces przetwarzania i analizy danych natychmiast po ich wygenerowaniu, umożliwiający podejmowanie decyzji na podstawie aktualnych informacji [Analytics, 2022]. W przeciwieństwie do tradycyjnego przetwarzania wsadowego (ang. batch processing), które operuje na dużych zbiorach historycznych danych, analiza w czasie rzeczywistym koncentruje się na przetwarzaniu strumieni danych (ang. stream processing) z minimalnymi opóźnieniami.

2.1.1 Modele przetwarzania danych

W literaturze wyróżnia się trzy główne modele przetwarzania danych [Processing, 2022]:

- **Przetwarzanie wsadowe (ang. batch processing)** - dane są gromadzone w pakiety i przetwarzane periodycznie, np. raz dziennie lub raz na tydzień. Model ten charakteryzuje się wysoką przepustowością, ale dużymi opóźnieniami.
- **Przetwarzanie mikrowsadowe (ang. micro-batch processing)** - dane są przetwarzane w małych pakietach, z częstotliwością rzędu sekund lub minut. Jest to kompromis między przetwarzaniem wsadowym a strumieniowym.
- **Przetwarzanie strumieniowe (ang. stream processing)** - dane są przetwarzane natychmiast po ich wygenerowaniu, indywidualnie lub w bardzo małych grupach. Model ten charakteryzuje się niskimi opóźnieniami, ale często niższą przepustowością.

W kontekście analizy danych z systemów wieloczujnikowych, szczególnie istotne jest przetwarzanie strumieniowe, które umożliwia szybką reakcję na zmieniające się warunki procesu produkcyjnego.

2.1.2 Wyzwania w analizie danych w czasie rzeczywistym

Analiza danych w czasie rzeczywistym, choć oferuje wiele korzyści, stawia przed projektantami systemów szereg istotnych wyzwań technicznych. Według **?**, kluczowe wyzwania obejmują:

- **Duża objętość i wysoka prędkość danych** - systemy czasu rzeczywistego muszą obsługiwać ogromne ilości danych napływających z wysoką prędkością. Wymaga to wydajnej infrastruktury zdolnej do przetwarzania danych z minimalnym opóźnieniem. Rozwiązaniem jest wykorzystanie systemów rozproszonych, optymalizacja algorytmów oraz efektywne zarządzanie zasobami.
- **Wymóg niskich opóźnień** - aplikacje czasu rzeczywistego wymagają niemal natychmiastowego przetwarzania danych i generowania odpowiedzi. Opóźnienia mogą prowadzić do nieaktualnych wyników i nieprawidłowych decyzji. Kluczowe jest zastosowanie przetwarzania w pamięci, optymalizacji zapytań i minimalizacji operacji wejścia/wyjścia.

- **Spójność i dokładność danych** - utrzymanie spójności danych podczas ich przetwarzania jest krytyczne dla poprawności analizy. Systemy muszą zapewnić dokładną semantykę przetwarzania (exactly-once processing) i obsługę nieporządku czasowego w danych.
- **Odporność na awarie i niezawodność** - systemy analizy czasu rzeczywistego muszą działać nieprzerwanie, nawet w przypadku awarii poszczególnych komponentów. Wymagane są mechanizmy automatycznego odzyskiwania po awarii, replikacji danych i równoważenia obciążenia.
- **Przetwarzanie złożonych zdarzeń** - wykrywanie wzorców i korelacji w strumieniach danych wymaga zaawansowanych technik przetwarzania zdarzeń. Konieczne jest zastosowanie algorytmów CEP (Complex Event Processing) zdolnych do identyfikacji istotnych wzorców w czasie rzeczywistym.
- **Integracja z istniejącymi systemami** - nowe rozwiązania muszą współpracować z istniejącą infrastrukturą IT. Wymaga to wykorzystania standardowych interfejsów, brokerów wiadomości i metod transformacji danych.
- **Skalowalność** - systemy powinny płynnie dostosowywać się do zmieniających się wolumenów danych i wzorców ruchu. Kluczowe jest projektowanie architektury umożliwiającej poziome skalowanie oraz elastyczne przydzielanie zasobów.
- **Bezpieczeństwo i prywatność** - ochrona wrażliwych danych podczas przetwarzania w czasie rzeczywistym stanowi istotne wyzwanie. Systemy muszą implementować szyfrowanie, kontrolę dostępu oraz mechanizmy audytu, zapewniając jednocześnie zgodność z przepisami dotyczącymi ochrony danych.

Skuteczne rozwiązanie tych wyzwań wymaga starannego projektowania architektury systemu, doboru odpowiednich technologii oraz uwzględnienia specyficznych wymagań aplikacji **?**. W kolejnych rozdziałach omówione zostaną narzędzia i techniki, które pozwalają sprostać tym wyzwaniom.

2.2 Systemy wieloczujnikowe w przemyśle

Systemy wieloczujnikowe to zbiory urządzeń pomiarowych, które monitorują różne parametry procesów przemysłowych [Systems, 2022]. W nowoczesnych zakładach produkcyjnych, systemy te generują ogromne ilości danych, które odpowiednio przetworzone mogą dostarczyć cennych informacji na temat stanu procesu.

2.2.1 Rodzaje czujników przemysłowych

W przemyśle chemicznym, w tym w procesie syntezy amoniaku, wykorzystuje się różne rodzaje czujników:

- **Czujniki temperatury** - monitorują temperaturę w różnych punktach instalacji
- **Czujniki ciśnienia** - mierzą ciśnienie gazu lub cieczy w instalacji
- **Czujniki przepływu** - monitorują przepływ materiałów przez instalację
- **Czujniki składu gazu** - analizują skład mieszanin gazowych, np. zawartość wodoru, azotu, amoniaku, tlenu czy dwutlenku węgla

- **Czujniki drgań i hałasu** - monitorują drgania i hałas generowane przez urządzenia, co może wskazywać na potencjalne problemy
- i inne

2.2.2 Architektura systemów wieloczujnikowych

Tradycyjne systemy wieloczujnikowe opierają się na architekturze trójwarstwowej:

- **Warstwa czujników** - obejmuje czujniki fizyczne wraz z konwerterami analogowo-cyfrowymi
- **Warstwa komunikacji** - odpowiada za przesyłanie danych z czujników do warstwy przetwarzania
- **Warstwa przetwarzania** - przetwarza dane z czujników i udostępnia je użytkownikom lub innym systemom

W nowoczesnych systemach, warstwa komunikacji często wykorzystuje technologie IoT, takie jak MQTT, AMQP czy OPC UA, a warstwa przetwarzania opiera się na mechanizmach przetwarzania strumieniowego, takich jak Apache Kafka czy Apache Flink.

2.3 Architektura mikroserwisowa

Architektura mikroserwisowa to podejście do tworzenia aplikacji jako zbioru luźno powiązanych, małych, autonomicznych usług, komunikujących się ze sobą za pomocą mechanizmów, takich jak protokół HTTP [Microservices, 2022]. Każdy mikroserwis realizuje określoną funkcjonalność biznesową i może być rozwijany, wdrażany i skalowany niezależnie od innych usług.

2.3.1 Zalety architektury mikroserwisowej

Architektura mikroserwisowa oferuje szereg zalet w kontekście systemów do analizy danych w czasie rzeczywistym [Benefits, 2022b]:

- **Skalowalność** - możliwość niezależnego skalowania poszczególnych usług w zależności od obciążenia
- **Odporność na awarie** - awaria jednego mikroserwisu nie powoduje awarii całego systemu
- **Elastyczność technologiczna** - możliwość wykorzystania różnych technologii i języków programowania w różnych mikroserwisach
- **Szybsze wdrażanie** - możliwość niezależnego wdrażania poszczególnych mikroserwisów
- **Łatwiejsze zarządzanie kodem** - mniejsze, bardziej zrozumiałe bazy kodu dla poszczególnych usług

2.3.2 Wyzwania architektury mikroserwisowej

Architektura mikroserwisowa stawia również pewne wyzwania [Challenges, 2022]:

- **Złożoność operacyjna** - zarządzanie wieloma niezależnymi usługami może być skomplikowane
- **Spójność danych** - trudności w utrzymaniu spójności danych między różnymi mikroserwisami
- **Koszty komunikacji sieciowej** - komunikacja między mikroserwisami wprowadza dodatkowe opóźnienia
- **Testowanie end-to-end** - trudności w testowaniu całego systemu złożonego z wielu niezależnych usług

2.4 Kubernetes jako platforma orkiestracji kontenerów

Kubernetes to otwarta platforma do orkiestracji kontenerów, która automatyzuje wdrażanie, skalowanie i zarządzanie aplikacjami kontenerowymi [Kubernetes, 2022]. Powstała jako projekt Google, obecnie rozwijana przez Cloud Native Computing Foundation (CNCF).

2.4.1 Podstawowe koncepcje Kubernetes

Kubernetes operuje na kilku kluczowych koncepcjach [Concepts, 2022]:

- **Pod** - najmniejsza jednostka wdrożeniowa w Kubernetes, składająca się z jednego lub więcej kontenerów
- **Deployment** - określa pożądany stan podów, umożliwiając ich skalowanie i aktualizacje
- **Service** - abstrakcja, która definiuje logiczny zestaw podów i politykę dostępu do nich
- **Ingress** - zarządza zewnętrznym dostępem do usług w klastrze
- **ConfigMap** i **Secret** - mechanizmy do przechowywania konfiguracji i tajnych danych
- **Namespace** - mechanizm do izolacji zasobów w klastrze

2.4.2 Zalety Kubernetes w kontekście analizy danych w czasie rzeczywistym

Kubernetes oferuje szereg zalet w kontekście rozproszonych systemów [Benefits, 2022a]:

- **Automatyczne skalowanie** - możliwość automatycznego dostosowywania liczby replik usług w zależności od obciążenia
- **Samonaprawianie** - automatyczne ponowne uruchamianie podów doznających awarii
- **Równoważenie obciążenia** - równomierne rozłożenie ruchu między replikami usług
- **Aktualizacje bez przestojów** - możliwość aktualizacji usług bez przerywania ich działania
- **Deklaratywna konfiguracja** - definiowanie pożądanego stanu systemu, a nie kroków do jego osiągnięcia

2.5 Narzędzia Big Data do przetwarzania strumieniowego

W kontekście analizy danych w czasie rzeczywistym, szczególnie istotne są narzędzia Big Data do przetwarzania strumieniowego. Poniżej omówiono kilka kluczowych technologii wykorzystywanych w tym obszarze.

2.5.1 Apache Kafka

Apache Kafka to rozproszona platforma do przetwarzania strumieniowego, opracowana przez LinkedIn, obecnie rozwijana jako projekt Apache Software Foundation [Kafka, 2022]. Kafka oferuje następujące funkcjonalności:

- **Wysoka przepustowość** - możliwość obsługi milionów wiadomości na sekundę
- **Trwałość danych** - dane są przechowywane na dysku i replikowane między brokerami
- **Skalowalność** - łatwe skalowanie poziome przez dodawanie nowych brokerów
- **Mechanizm partycjonowania** - umożliwia równoległe przetwarzanie danych
- **Gwarancje dostarczania** - co najmniej raz, co najwyżej raz lub dokładnie raz

2.5.2 Kafka Streams

Kafka Streams to biblioteka przetwarzania strumieniowego, zintegrowana z Apache Kafka [Streams, 2022]. Oferuje następujące funkcjonalności:

- **Przetwarzanie rekord po rekordzie** - minimalne opóźnienia przetwarzania
- **Operacje stanowe i bezstanowe** - możliwość agregacji danych w czasie
- **Okna czasowe** - przetwarzanie danych w zdefiniowanych oknach czasowych
- **Łączenie strumieni** - możliwość łączenia danych z różnych strumieni
- **Dokładnie raz** - gwarancje przetwarzania dokładnie raz, eliminujące duplikaty i utratę danych

2.5.3 Apache Flink

Apache Flink to framework przetwarzania strumieniowego, oferujący funkcjonalności podobne do Kafka Streams, ale jako oddzielna platforma [Flink, 2022]. Flink charakteryzuje się:

- **Niskimi opóźnieniami** - przetwarzanie rekord po rekordzie z minimalnymi opóźnieniami
- **Wysoką przepustowością** - efektywne przetwarzanie dużych wolumenów danych
- **Dokładnie raz** - gwarancje przetwarzania dokładnie raz
- **Zaawansowanym zarządzaniem stanem** - efektywne przechowywanie i dostęp do stanu przetwarzania
- **Obsługą czasu zdarzeń** - możliwość przetwarzania danych na podstawie czasu, w którym zdarzenia zostały wygenerowane

2.5.4 Apache Spark Streaming

Apache Spark Streaming to moduł przetwarzania strumieniowego platformy Apache Spark [Streaming, 2022]. Opiera się na modelu mikrowsadowym, gdzie dane są przetwarzane w małych pakietach. Spark Streaming oferuje:

- **Integrację z ekosystemem Spark** - możliwość wykorzystania bibliotek Spark do analizy danych i uczenia maszynowego
- **Wysoką przepustowość** - efektywne przetwarzanie dużych wolumenów danych
- **Odporność na awarie** - automatyczne odtwarzanie stanu po awarii
- **Łatwe skalowanie** - możliwość łatwego skalowania przetwarzania przez dodawanie węzłów

2.6 Istniejące rozwiązania i ich ograniczenia

W literaturze i praktyce przemysłowej istnieje szereg rozwiązań do analizy danych z systemów wieloczujnikowych. Poniżej omówiono kilka z nich, wraz z ich ograniczeniami.

2.6.1 Tradycyjne systemy SCADA

Systemy SCADA (Supervisory Control and Data Acquisition) to tradycyjne rozwiązania do monitorowania i kontroli procesów przemysłowych [SCADA, 2022]. Mimo popularności, systemy te mają pewne ograniczenia w kontekście analizy danych w czasie rzeczywistym:

- **Ograniczona skalowalność** - trudności w obsłudze dużej liczby czujników i strumieni danych
- **Monolityczna architektura** - utrudnia elastyczne rozwijanie i modyfikowanie systemu
- **Ograniczone możliwości analityczne** - często koncentrują się na wizualizacji danych, a nie ich głębokiej analizie
- **Wysokie koszty licencji** - komercyjne systemy SCADA często wiążą się z wysokimi kosztami licencji

2.6.2 Platformy IoT w chmurze

Platformy IoT w chmurze, takie jak AWS IoT, Azure IoT Hub czy Google Cloud IoT Core, oferują zaawansowane możliwości analizy danych z urządzeń IoT [IoT, 2022]. Mimo to, mają pewne ograniczenia:

- **Zależność od dostawcy chmury** - trudności w migracji między różnymi dostawcami
- **Koszty transferu danych** - wysokie koszty przy dużym wolumenie danych
- **Opóźnienia sieciowe** - potencjalne opóźnienia związane z przesyłaniem danych do chmury
- **Ograniczone możliwości dostosowania** - platformy chmurowe oferują określony zestaw usług, które mogą nie spełniać wszystkich wymagań

2.6.3 Open source rozwiązania do analizy danych w czasie rzeczywistym

Istnieje szereg open source rozwiązań do analizy danych w czasie rzeczywistym, takich jak Apache NiFi, Apache Druid czy InfluxDB . Mimo ich zalet, mają również pewne ograniczenia:

- **Złożoność wdrożenia** - konfiguracja i wdrożenie mogą być skomplikowane
- **Ograniczone wsparcie** - wsparcie techniczne może być ograniczone w porównaniu do rozwiązań komercyjnych
- **Konieczność integracji wielu narzędzi** - często wymagają integracji wielu narzędzi, co zwiększa złożoność systemu

Podsumowując, istniejące rozwiązania do analizy danych z systemów wieloczujnikowych mają pewne ograniczenia, które uzasadniają potrzebę opracowania nowego systemu, opartego na architekturze mikroserwisowej, klastrze Kubernetes i narzędziach Big Data.

3 Projekt systemu przetwarzania danych w czasie rzeczywistym

W niniejszym rozdziale przedstawiono projekt systemu do analizy danych w czasie rzeczywistym z systemów wieloczujnikowych, opartego na klastrze Kubernetes oraz narzędziach Big Data. Omówiono wymagania systemu, architekturę, model danych oraz przepływ danych.

3.1 Wymagania systemu

Na podstawie analizy literatury oraz istniejących rozwiązań, zidentyfikowano następujące wymagania dla projektowanego systemu:

3.1.1 Wymagania funkcjonalne

1. **Pozyskiwanie danych** - system powinien umożliwiać pozyskiwanie danych z różnych typów czujników
2. **Przetwarzanie danych** - system powinien umożliwiać przetwarzanie danych w czasie rzeczywistym, w tym:
   - Filtrację danych
   - Agregację danych w różnych oknach czasowych
   - Wykrywanie anomalii
   - Przewidywanie awarii
   - Korelację danych z wielu czujników
3. **Wizualizacja danych** - system powinien umożliwiać wizualizację danych w czasie rzeczywistym, w tym:
   - Wykresy liniowe i słupkowe
   - Wskaźniki i mierniki
   - Alerty i powiadomienia
4. **Raportowanie** - system powinien umożliwiać generowanie raportów, w tym:

3.1.2 Wymagania niefunkcjonalne

1. **Skalowalność** - system powinien umożliwiać:
   - Skalowanie horyzontalne (dodawanie nowych węzłów)
   - Automatyczne skalowanie w zależności od obciążenia
2. **Niezawodność** - system powinien charakteryzować się:
   - Odpornością na awarie pojedynczych komponentów
3. **Bezpieczeństwo** - system powinien zapewniać:

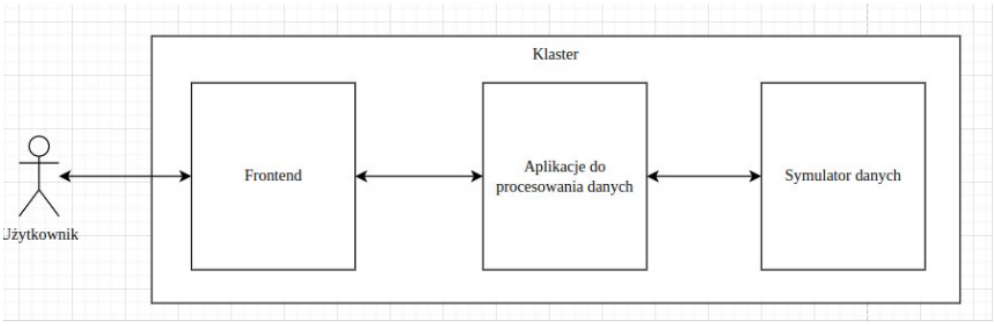
- Szyfrowanie danych w spoczynku i w transporcie
- Uwierzytelnianie i autoryzację użytkowników
- Audytowanie dostępu do danych

4. **Utrzymywalność** - system powinien charakteryzować się:

- Modułową architekturą
- Dobrą dokumentacją
- Łatwością rozszerzania i modyfikacji

3.2 Architektura systemu

Projektowany system opiera się na architekturze mikroserwisowej, wdrożonej na klastrze Kubernetes. Architektura składa się z kilku kluczowych warstw, przedstawionych na Rysunku 1.



Rysunek 1: Architektura systemu do analizy danych w czasie rzeczywistym

3.2.1 Warstwa pozyskiwania danych (symulatory czujników)

Warstwa pozyskiwania danych odpowiada za zbieranie danych z czujników i ich wstępne przetworzenie. W projektowanym systemie, dane są generowane przez symulatory czujników implementowane jako funkcje AWS Lambda. Symulatory te generują dane symulujące odczyty z różnych typów czujników.

Główne komponenty tej warstwy to:

Dane generowane przez symulatory są formatowane jako wiadomości JSON i publikowane na tematy SNS, a następnie przenoszone do kolejek SQS, które stanowią interfejs między AWS a klastrem Kubernetes.

3.2.2 Warstwa komunikacji (Apache Kafka)

Warstwa komunikacji odpowiada za odbieranie danych z warstwy pozyskiwania i ich dostarczanie do warstwy przetwarzania. W projektowanym systemie, warstwa ta opiera się na Apache Kafka, rozproszonej platformie do przetwarzania strumieniowego.

Apache Kafka zapewnia trwałość danych, wysoką przepustowość i skalowalność, co jest kluczowe w kontekście przetwarzania danych w czasie rzeczywistym z wielu czujników.

3.2.3 Warstwa przetwarzania (Kafka Streams)

Warstwa przetwarzania odpowiada za analizę danych w czasie rzeczywistym. W projektowanym systemie, warstwa ta opiera się na Kafka Streams, bibliotece przetwarzania strumieniowego zintegrowanej z Apache Kafka.

3.2.4 Warstwa składowania danych

Warstwa składowania danych odpowiada za przechowywanie przetworzonych danych do dalszej analizy i wizualizacji. W projektowanym systemie, warstwa ta składa się z dwóch głównych komponentów:

- **PostgreSQL** - relacyjna baza danych, przechowująca ustrukturyzowane dane, takie jak odczyty czujników, metadane czy informacje o użytkownikach
- **Elasticsearch** - baza danych NoSQL, umożliwiająca szybkie wyszukiwanie i analizę danych, szczególnie przydatna w kontekście wykrywania anomalii i analizy trendów

3.2.5 Warstwa usług i API

Warstwa usług i API odpowiada za udostępnianie danych i funkcjonalności systemu zewnętrznym aplikacjom i użytkownikom. W projektowanym systemie, warstwa ta składa się z kilku mikroserwisów:

- **Data Service** - mikroserwis udostępniający API do dostępu do danych z czujników, umożliwiający tworzenie i pobieranie raportów
- **Users Service** - mikroserwis zarządzający użytkownikami i uwierzytelnianiem, wykorzystujący Keycloak jako system zarządzania tożsamością
- **Front Service** - mikroserwis pełniący rolę API Gateway, który ekspozuje API innych mikroserwisów na zewnątrz klastra

Wszystkie mikroserwisy są implementowane jako aplikacje Spring Boot, co zapewnia łatwość rozwoju, testowania i wdrażania.

3.2.6 Warstwa prezentacji

Warstwa prezentacji odpowiada za wizualizację danych i interakcję z użytkownikami. W projektowanym systemie, warstwa ta składa się z dashboardu, który wyświetla dane w formie wykresów, tabel i widżetów.

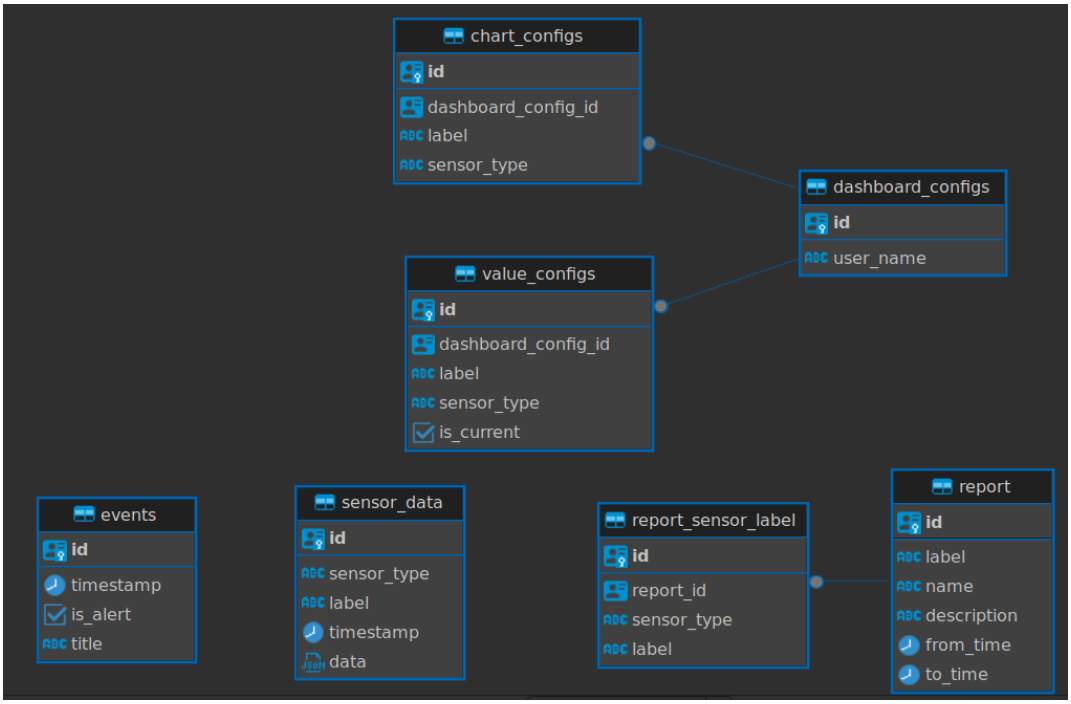
Główne funkcjonalności dashboardu to:

- **Wykresy liniowe** - wizualizacja zmian parametrów w czasie
- **Wskaźniki i mierniki** - wizualizacja aktualnych wartości parametrów
- **Alerty** - powiadomienia o anomaliach i potencjalnych awariach
- **Raporty** - generowanie i przeglądanie raportów

Dashboard jest implementowany jako aplikacja webowa, wykorzystująca React i biblioteki wizualizacji danych.

3.3 Model danych

Model danych projektowanego systemu składa się z kilku kluczowych encji, przedstawionych na Rysunku 2.



Rysunek 2: Model danych systemu

3.3.1 Dane z czujników

Dane z czujników są modelowane jako strumienie zdarzeń, gdzie każde zdarzenie zawiera:

- **Typ czujnika** - rodzaj czujnika (temperatura, ciśnienie, przepływ, skład gazu, drgania)
- **Znacznik lokalizacji** - miejsce, w którym znajduje się czujnik
- **Znacznik czasu** - czas, w którym dokonano pomiaru
- **Wartość** - wartość pomiaru
- **Jednostka** - jednostka, w której wyrażona jest wartość

Dane te są serializowane przy użyciu Apache Avro, co zapewnia efektywne kodowanie i dekodowanie wiadomości.

3.3.2 Przetworzone dane

Przetworzone dane są wynikiem analizy danych surowych i zgodnie z modelem danych przedstawionym na Rysunku 2 obejmują:

- **Agregacje (Aggregations)** - statystyki obliczane w różnych oknach czasowych na podstawie danych surowych:

  - *measurement_id* - unikalny identyfikator pomiaru
  - *sensor_id* - identyfikator czujnika, którego dotyczą agregacje
  - *timestamp* - znacznik czasu agregacji
  - *window_size* - rozmiar okna czasowego (np. 1 minuta, 5 minut, 1 godzina)
  - *avg_value* - średnia wartość w oknie czasowym
  - *min_value* - minimalna wartość w oknie czasowym
  - *max_value* - maksymalna wartość w oknie czasowym
  - *std_dev* - odchylenie standardowe wartości w oknie czasowym

- **Anomalie (Anomalies)** - wykryte odstępstwa od normalnego zachowania czujników:
  - *anomaly_id* - unikalny identyfikator anomalii
  - *sensor_id* - identyfikator czujnika, na którym wykryto anomalię
  - *timestamp* - czas wykrycia anomalii
  - *type* - typ anomalii (np. spike, drift, stuck value)
  - *severity* - poziom ważności (niski, średni, wysoki, krytyczny)
  - *description* - tekstowy opis anomalii
  - *actual_value* - rzeczywista wartość, która spowodowała anomalię
  - *expected_value* - oczekiwana wartość na podstawie modelu

- **Predykcje (Predictions)** - przewidywane wartości parametrów lub prawdopodobieństwa awarii:
  - *prediction_id* - unikalny identyfikator predykcji
  - *sensor_id* - identyfikator czujnika, którego dotyczy predykcja
  - *timestamp* - czas utworzenia predykcji
  - *prediction_horizon* - horyzont czasowy predykcji (np. 1 minuta, 1 godzina, 1 dzień)
  - *predicted_value* - przewidywana wartość
  - *confidence* - poziom pewności predykcji (0-100
  - *model_type* - typ modelu użytego do predykcji

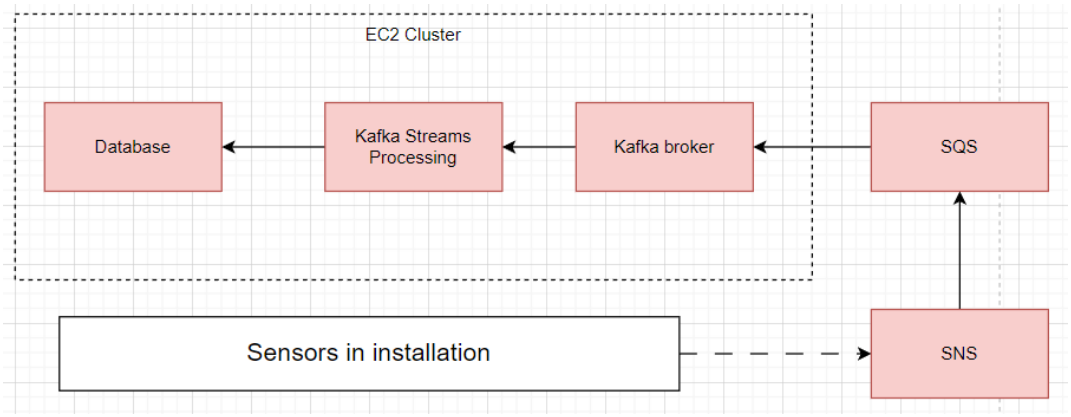
- **Korelacje (Correlations)** - zidentyfikowane zależności między różnymi parametrami:
  - *correlation_id* - unikalny identyfikator korelacji
  - *source_sensor_id* - identyfikator pierwszego czujnika
  - *target_sensor_id* - identyfikator drugiego czujnika
  - *timestamp* - czas obliczenia korelacji
  - *window_size* - rozmiar okna czasowego użytego do obliczenia korelacji
  - *correlation_coefficient* - współczynnik korelacji (-1 do 1)
  - *time_lag* - opóźnienie czasowe między sygnałami

– *significance* - istotność statystyczna korelacji

Wszystkie te dane są przechowywane w bazie danych PostgreSQL, w tabelach odpowiadających poszczególnym typom danych. Relacje między tabelami są zdefiniowane przez klucze obce, co umożliwia efektywne wykonywanie złożonych zapytań analitycznych.

3.4 Przepływ danych w systemie

Przepływ danych w projektowanym systemie obejmuje kilka etapów, przedstawionych na Rysunku 3.



Rysunek 3: Przepływ danych w systemie

3.4.1 Generowanie danych

Proces rozpoczyna się od generowania danych przez symulatory czujników. Symulatory te są implementowane jako funkcje AWS Lambda, które są wyzwalane periodycznie przez AWS Step Functions. Każdy symulator generuje dane symulujące odczyty z określonego typu czujnika, umieszczonego w określonym miejscu instalacji do syntezy amoniaku.

Dane są generowane w formacie JSON i zawierają informacje takie jak identyfikator czujnika, typ, lokalizacja, znacznik czasu, wartość i jednostka. Dane te są następnie publikowane na temat SNS (Simple Notification Service), który działa jako punkt dystrybucji dla wielu subskrybentów.

3.4.2 Przesyłanie danych do klastra

Dane opublikowane na temacie SNS są automatycznie dostarczane do kolejek SQS (Simple Queue Service), które są subskrybentami tematu. Kolejki SQS działają jako bufor między AWS a klastrem Kubernetes, zapewniając niezawodne dostarczanie wiadomości, nawet w przypadku tymczasowej niedostępności klastra.

W klastrze Kubernetes działa mikroserwis SQS-Kafka-Forwarder, który regularnie odpytuje kolejki SQS i pobiera nowe wiadomości. Wiadomości te są następnie deserializowane i publikowane na odpowiednie tematy Kafka, w zależności od typu czujnika i lokalizacji.

3.4.3 Przetwarzanie strumieni danych

Dane opublikowane na tematach Kafka są przetwarzane przez mikroserwis Kafka-Data-Processor, który wykorzystuje Kafka Streams do analizy danych w czasie rzeczywistym. Przetwarzanie obejmuje:

- **Filtrację** - usuwanie nieprawidłowych lub niekompletnych danych
- **Transformację** - przekształcanie danych do odpowiedniego formatu
- **Agregację** - obliczanie statystyk w różnych oknach czasowych
- **Analizę** - wykrywanie anomalii, przewidywanie awarii, korelację danych

Wyniki przetwarzania są publikowane na nowe tematy Kafka, które zawierają przetworzone dane.

3.4.4 Zapisywanie danych do bazy danych

Przetworzone dane są konsumowane przez mikroserwis Kafka-DB-Forwarder, który zapisuje je do baz danych. Dane strukturalne, takie jak odczyty czujników, agregacje i metadane, są zapisywane do bazy danych PostgreSQL, natomiast dane semi-strukturalne, takie jak logi i zdarzenia, są zapisywane do Elasticsearch.

Zapisywanie danych do baz danych umożliwia ich późniejszą analizę, raportowanie i wizualizację, a także zapewnia trwałość danych.

3.4.5 Udostępnianie danych przez API

Zapisane dane są udostępniane przez mikroserwis Data-Service, który ekspozuje REST API do tworzenia i pobierania raportów. API to umożliwia:

- **Pobieranie danych surowych** - dostęp do nieprzetworzonych odczytów czujników
- **Pobieranie danych przetworzonych** - dostęp do agregacji, wykrytych anomalii, predykcji
- **Tworzenie raportów** - generowanie raportów na podstawie zadanych kryteriów
- **Pobieranie raportów** - dostęp do wcześniej wygenerowanych raportów

API jest zabezpieczone mechanizmami uwierzytelniania i autoryzacji, implementowanymi przez mikroserwis Users-Service, który korzysta z Keycloak.

3.4.6 Wizualizacja danych

Ostatnim etapem przepływu danych jest ich wizualizacja na dashboardzie. Dashboard komunikuje się z API, pobierając dane do wyświetlenia, a następnie prezentuje je w formie wykresów, tabel i widżetów.

Dashboard umożliwia interaktywną eksplorację danych, filtrowanie, sortowanie i eksport wyników. Ponadto, dashboard może wyświetlać alerty i powiadomienia o wykrytych anomaliach i potencjalnych awariach.

Przepływ danych w projektowanym systemie zapewnia efektywne pozyskiwanie, przetwarzanie i analizę danych w czasie rzeczywistym, co jest kluczowe w kontekście monitorowania i optymalizacji procesu syntezy amoniaku.

4 Implementacja systemu przetwarzania danych w czasie rzeczywistym

W niniejszym rozdziale przedstawiono szczegóły implementacji systemu do analizy danych w czasie rzeczywistym z systemów wieloczujnikowych, zgodnie z projektem opisanym w poprzednim rozdziale.

4.1 Wykorzystane technologie

Kluczowe technologie wykorzystane w implementacji to:

4.1.1 Języki programowania

- **Java 21** - główny język programowania wykorzystany do implementacji mikroserwisów przetwarzających dane.
- **Python 3.9** - język wykorzystany do implementacji symulatorów czujników i skryptów pomocniczych.
- **TypeScript** - język wykorzystany do implementacji interfejsu użytkownika.

4.1.2 Frameworki i biblioteki

- **Spring Boot 3.4** - framework wykorzystany do implementacji mikroserwisów.
- **Kafka Streams 3.4** - biblioteka wykorzystana do przetwarzania strumieniowego.
- **Spring Cloud** - zestaw narzędzi do budowy aplikacji chmurowych.
- **React 18** - biblioteka JavaScript wykorzystana do budowy interfejsu użytkownika.

4.1.3 Bazy danych i systemy magazynowania

- **Apache Kafka 3.4** - platforma do przetwarzania strumieniowego.
- **PostgreSQL 15** - relacyjna baza danych.
- **Elasticsearch 8.7** - baza danych NoSQL do wyszukiwania i analizy.
- **Redis 7.0** - baza danych in-memory wykorzystywana jako cache.

4.1.4 Infrastruktura i orkiestracja

- **Kubernetes 1.26** - platforma do orkiestracji kontenerów.
- **Docker** - platforma konteneryzacji.
- **Helm 3** - menedżer pakietów dla Kubernetes.
- **AWS** - chmura obliczeniowa wykorzystana do hostowania usług zewnętrznych.
- **Terraform** - narzędzie do zarządzania infrastrukturą jako kod.

4.2 Implementacja symulatora czujników

4.2.1 Architektura symulatora

Architektura symulatora składa się z kilku kluczowych komponentów:

- **AWS Step Functions** - usługa, która koordynuje wykonywanie funkcji Lambda, zarządzając przepływem pracy i częstotliwością generowania danych.
- **AWS Lambda** - usługa bezserwerowa, która wykonuje kod symulatora.
- **AWS SNS** - usługa powiadomień, która działa jako punkt dystrybucji dla wygenerowanych danych.
- **AWS SQS** - usługa kolejek, która buforuje wiadomości przed ich przetworzeniem przez system.

4.2.2 Implementacja funkcji Lambda

Funkcje Lambda zostały zaimplementowane w języku Python 3.9. Każda funkcja generuje dane dla określonego typu czujnika, symulując jego zachowanie w różnych warunkach pracy. Symulator uwzględnia specyfikę każdego typu czujnika i jego charakterystykę w kontekście procesu syntezy amoniaku.

Dla każdego typu czujnika zaimplementowano logikę generowania realistycznych danych, uwzględniającą:

- Normalne wahania wartości w zakresie typowym dla danego parametru
- Możliwość wystąpienia anomalii z określonym prawdopodobieństwem
- Korelacje między różnymi parametrami procesu
- Symulację zakłóceń i szumów pomiarowych

5 Zastosowania praktyczne

W niniejszym rozdziale przedstawiono praktyczne zastosowania opracowanego systemu analizy danych w czasie rzeczywistym z systemów wieloczujnikowych w różnych sektorach przemysłu. Omówiono korzyści ekonomiczne i operacyjne wynikające z wdrożenia systemu oraz wyzwania napotkane podczas implementacji w rzeczywistych środowiskach produkcyjnych.

5.1 Przypadki użycia w przemyśle

Opracowany system znalazł zastosowanie w kilku gałęziach przemysłu, gdzie analiza danych w czasie rzeczywistym przynosi wymierne korzyści.

5.1.1 Przemysł motoryzacyjny

W fabryce produkującej komponenty automotive system został wdrożony do monitorowania i optymalizacji linii produkcyjnej:

- **Monitorowanie stanu maszyn** - system zbiera dane z czujników zainstalowanych na kluczowych maszynach produkcyjnych (prasy, roboty spawalnicze, linie montażowe),
- analizując w czasie rzeczywistym parametry takie jak temperatura, wibracje, ciśnienie i zużycie energii.
- **Predykcja awarii** - na podstawie historycznych danych o awariach oraz wzorców anomalii system przewiduje potencjalne usterki z wyprzedzeniem 24-48 godzin,
- umożliwiając zaplanowanie konserwacji prewencyjnej bez zakłócania harmonogramu produkcji.
- **Kontrola jakości** - analiza danych z czujników w procesie produkcyjnym pozwala na wczesne wykrywanie odchyleń parametrów, które mogłyby prowadzić do defektów
- produktów. System automatycznie alarmuje operatorów, gdy parametry procesu zbliżają się do wartości granicznych.

Po sześciomiesięcznym okresie użytkowania systemu zanotowano 37% redukcję nieplanowanych przestojów linii produkcyjnej oraz 12% zmniejszenie ilości wadliwych produktów.

5.1.2 Przemysł energetyczny

W elektrowni wykorzystującej odnawialne źródła energii (farma wiatrowa i instalacja fotowoltaiczna) system znalazł zastosowanie do:

- **Optymalizacji produkcji energii** - analiza danych z czujników meteorologicznych, turbinowych oraz z sieci dystrybucji pozwala na dynamiczne dostosowanie
- parametrów pracy turbin wiatrowych i optymalne ukierunkowanie paneli fotowoltaicznych.
- **Predykcji produkcji energii** - na podstawie danych historycznych oraz prognoz pogody system przewiduje produkcję energii z wyprzedzeniem 24-72 godzin, co

- umożliwia efektywne zarządzanie magazynami energii i planowanie dostaw do sieci.
- **Wczesnego wykrywania usterek** - system monitoruje parametry pracy każdej turbiny wiatrowej i sekcji paneli fotowoltaicznych, wykrywając nietypowe wzorce
- wskazujące na rozwijające się usterki (np. zużycie łożysk w turbinach czy degradację ogniw fotowoltaicznych).

Wdrożenie systemu przyczyniło się do 8% wzrostu efektywności produkcji energii oraz 23% redukcji kosztów konserwacji poprzez przejście z modelu konserwacji okresowej na model konserwacji prewencyjnej oparty na rzeczywistym stanie urządzeń.

5.1.3 Przemysł spożywczy

W zakładzie przetwórstwa spożywczego system wykorzystywany jest do:

- **Monitorowania łańcucha chłodniczego** - czujniki temperatury i wilgotności rozmieszczone w magazynach, komorach chłodniczych i podczas transportu dostarczają danych, które są analizowane w czasie rzeczywistym, zapewniając utrzymanie optymalnych warunków przechowywania.
- **Kontroli parametrów procesów produkcyjnych** - w procesach takich jak pasteryzacja, fermentacja czy suszenie system monitoruje parametry krytyczne dla jakości i bezpieczeństwa produktów, natychmiast alarmując o odchyleniach.
- **Optymalizacji zużycia mediów** - analiza korelacji między zużyciem energii, wody i innych mediów a parametrami produkcji pozwala na identyfikację nieefektywności i optymalizację procesów.

Implementacja systemu przyniosła 15% redukcję strat produktów spowodowanych niewłaściwymi warunkami przechowywania oraz 9% oszczędności na zużyciu energii.

5.1.4 Smart City i infrastruktura miejska

W projekcie pilotażowym dla jednego z polskich miast system został wdrożony do:

- **Monitorowania jakości powietrza** - sieć czujników rozproszona po całym mieście dostarcza danych o stężeniu PM2.5, PM10, CO2, NOx i innych zanieczyszczeń.
- System analizuje te dane w czasie rzeczywistym, tworząc dynamiczne mapy jakości powietrza i prognozując rozwój sytuacji.
- **Zarządzania ruchem drogowym** - dane z kamer, czujników natężenia ruchu i sygnalizacji świetlnej są analizowane w celu optymalizacji sterowania ruchem, redukcji korków i emisji spalin.
- **Monitorowania infrastruktury krytycznej** - czujniki na mostach, wiaduktach, przepompowniach i innych elementach infrastruktury krytycznej dostarczają
- danych o stanie tych obiektów, umożliwiając wczesne wykrywanie symptomów potencjalnych awarii.

Wykorzystanie systemu w zarządzaniu ruchem miejskim przyczyniło się do 17% redukcji średniego czasu przejazdu przez centrum miasta w godzinach szczytu oraz 11% zmniejszenia emisji CO2 z transportu miejskiego.

5.2 Perspektywy rozwoju zastosowań praktycznych

Dotychczasowe doświadczenia z wdrożeń systemu oraz analiza trendów technologicznych i biznesowych pozwalają na zidentyfikowanie kluczowych kierunków rozwoju zastosowań praktycznych w przyszłości.

5.2.1 Kierunki rozwoju technologicznego

- **Edge computing** - przesunięcie części przetwarzania danych na urządzenia brzegowe, co zmniejsza opóźnienia, redukuje zapotrzebowanie na przepustowość sieci i zwiększa odporność systemu na problemy z łącznością.
- **Sztuczna inteligencja i uczenie maszynowe** - dalszy rozwój algorytmów AI/ML do bardziej zaawansowanej analizy danych, umożliwiającej wykrywanie subtelnych wzorców i zależności, niedostrzegalnych dla tradycyjnych metod analitycznych.
- **Digital Twin** - integracja systemów analityki czasu rzeczywistego z cyfrowymi bliźniakami procesów i urządzeń, umożliwiająca symulacje "what-if" i optymalizację procesów w środowisku wirtualnym przed wdrożeniem zmian w rzeczywistości.
- **5G i IoT** - wykorzystanie sieci 5G do komunikacji z rozproszonymi czujnikami, co umożliwia monitoring urządzeń mobilnych i instalacji w trudno dostępnych lokalizacjach.

5.2.2 Nowe obszary zastosowań

- **Rolnictwo precyzyjne** - monitoring parametrów glebowych, pogodowych i stanu upraw w celu optymalizacji nawadniania, nawożenia i ochrony roślin.
- **Medycyna i opieka zdrowotna** - analiza danych z urządzeń medycznych i wearables w czasie rzeczywistym, umożliwiająca wczesne wykrywanie symptomów chorób i personalizację terapii.
- **Logistyka i łańcuchy dostaw** - monitoring w czasie rzeczywistym wszystkich elementów łańcucha dostaw, umożliwiający optymalizację tras, redukcję opóźnień i lepsze zarządzanie zapasami.
- **Autonomiczne pojazdy i drony** - przetwarzanie danych z wielu czujników (kamery, lidar, radar) w czasie rzeczywistym, umożliwiające autonomiczną nawigację i podejmowanie decyzji.
- **Zarządzanie kryzysowe** - analiza danych z różnych źródeł (czujniki, media społecznościowe, systemy monitoringu) w sytuacjach kryzysowych, wspomagająca koordynację działań służb ratunkowych.

Rozwój tych obszarów zastosowań będzie napędzany zarówno przez postęp technologiczny, jak i rosnącą świadomość korzyści płynących z analityki danych w czasie rzeczywistym wśród decydentów w różnych sektorach gospodarki.

6 Podsumowanie i wnioski

Spis rysunków

Spis tabel

Literatura

Analytics. Real-time analytics: Definition, examples, and use cases, 2022. URL https://www.databricks.com/glossary/real-time-analytics. [Accessed: 01/01/2024].

Benefits. Benefits of using kubernetes, 2022a. URL https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. [Accessed: 01/01/2024].

Benefits. Benefits of microservices architecture, 2022b. URL https://www.nginx.com/blog/introduction-to-microservices/. [Accessed: 01/01/2024].

Challenges. Challenges in microservices architecture, 2022. URL https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/. [Accessed: 01/01/2024].

Concepts. Kubernetes core concepts, 2022. URL https://kubernetes.io/docs/concepts/. [Accessed: 01/01/2024].

Flink. Apache flink documentation, 2022. URL https://flink.apache.org/docs/stable/. [Accessed: 01/01/2024].

IoT. Cloud iot platforms comparison, 2022. URL https://www.postscapes.com/internet-of-things-platforms/. [Accessed: 01/01/2024].

Kafka. Apache kafka documentation, 2022. URL https://kafka.apache.org/documentation/. [Accessed: 01/01/2024].

Kubernetes. Kubernetes documentation, 2022. URL https://kubernetes.io/docs/home/. [Accessed: 01/01/2024].

Microservices. Pattern: Microservice architecture, 2022. URL https://microservices.io/patterns/microservices.html. [Accessed: 01/01/2024].

Processing. Data processing models: Batch vs. stream processing, 2022. URL https://www.confluent.io/learn/batch-vs-real-time-data-processing/. [Accessed: 01/01/2024].

SCADA. Scada systems: What they are and how they work, 2022. URL https://www.inductiveautomation.com/resources/article/what-is-scada. [Accessed: 01/01/2024].

Streaming. Spark streaming documentation, 2022. URL https://spark.apache.org/docs/latest/streaming-programming-guide.html. [Accessed: 01/01/2024].

Streams. Kafka streams documentation, 2022. URL https://kafka.apache.org/documentation/streams/. [Accessed: 01/01/2024].

Systems. Multi-sensor systems in industrial applications, 2022. URL https://www.sciencedirect.com/science/article/pii/S2405896318332421. [Accessed: 01/01/2024].