O'REILLY®

Wydanie II

Przetwarzanie danych w dużej skali

Niezawodność, skalowalność i konserwacja systemów

Martin Kleppmann
Chris Riccomini

Helion

Tytuł oryginału: Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, 2nd Edition

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-289-3954-7

© 2026 Helion S.A.

Authorized Polish translation of the English edition of *Designing Data-Intensive Applications, 2E* ISBN 9791098119065 © 2026 Martin Kleppmann, Chris Riccomini.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/przed2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

ROZDZIAŁ 1.

Kompromisy w architekturze systemów danych

Nie ma idealnych rozwiązań, są tylko kompromisy. […]
Ale starasz się uzyskać najlepszy możliwy kompromis i to wszystko, na co możesz liczyć.

— Thomas Sowell, wywiad z Fredem Barnesem (2005)

Dane stanowią dziś ważny aspekt tworzenia większości aplikacji. Wraz z pojawieniem się aplikacji internetowych i mobilnych, modelu SaaS oraz usług chmurowych standardem stało się przechowywanie danych wielu użytkowników we współdzielonej infrastrukturze serwerowej. Dane otrzymywane na podstawie aktywności użytkowników i transakcji biznesowych oraz pochodzące z urządzeń i czujników muszą być gromadzone i udostępniane do analizy. Gdy użytkownicy korzystają z aplikacji, zarówno wczytują przechowywane dane, jak i generują nowe.

Małe ilości danych, które można przechowywać i przetwarzać na jednej maszynie, zazwyczaj nie sprawiają większych problemów. Jednak gdy wolumen danych lub liczba zapytań rośnie, konieczne staje się wprowadzenie modelu rozproszonego z wykorzystaniem wielu maszyn, co rodzi liczne wyzwania. Kiedy wymogi aplikacji stają się bardziej złożone, przechowywanie wszystkich danych w jednym systemie przestaje wystarczać. Konieczne może być wtedy połączenie wielu systemów do przechowywania lub przetwarzania danych, oferujących różne możliwości.

Aplikację nazywamy **intensywnie przetwarzającą dane** (ang. *data-intensive*), gdy zarządzanie danymi stanowi jedno z głównych wyzwań przy jej tworzeniu [1]. Podczas gdy w systemach *wymagających obliczeniowo* (ang. *compute-intensive*) wyzwaniem jest równoległe wykonywanie wysoce wymagających obliczeń, w aplikacjach intensywnie przetwarzających dane zwykle bardziej problematyczne są kwestie takie jak przechowywanie i przetwarzanie dużych wolumenów danych, zarządzanie zmianami w danych, zapewnienie spójności w obliczu awarii i współbieżności oraz utrzymanie wysokiej dostępności usług.

Aplikacje intensywnie przetwarzające dane są zwykle zbudowane ze standardowych cegiełek, które zapewniają często potrzebne mechanizmy. Na przykład wiele aplikacji musi:

- przechowywać dane, aby później dana aplikacja (lub inna) mogła je znaleźć (**baza danych**);
- zapamiętywać wyniki kosztownych operacji, aby przyspieszyć odczyt (**pamięć podręczna**);

Kup książkę Poleć książkę

- umożliwiać użytkownikom wyszukiwanie danych na podstawie słów kluczowych lub filtrów (**indeksy wyszukiwania**);
- przesyłać do innego procesu komunikat w celu jego asynchronicznej obsługi (**przetwarzanie strumieniowe**);
- okresowo przetwarzać duże ilości zakumulowanych danych (**przetwarzanie wsadowe**).

W trakcie budowania aplikacji zazwyczaj łączymy kilka systemów lub usług, takich jak bazy danych czy API, które są spajane ze sobą za pomocą kodu aplikacji. Jeśli wykorzystujesz systemy danych zgodnie z ich przeznaczeniem, proces ten może być całkiem prosty.

Jednak w bardziej ambitnych projektach mogą pojawiać się wyzwania. Istnieje wiele systemów bazodanowych o różnych charakterystykach, odpowiednich do różnych celów. Jak wybrać właściwy system? Istnieją też różne sposoby obsługi pamięci podręcznej, budowania indeksów wyszukiwania itd. Jak ocenić zalety i wady poszczególnych rozwiązań? Musisz ustalić, które narzędzia i podejścia najlepiej pasują do danego zadania. Gdy pojedyncze narzędzia okazują się niewystarczające, pamiętaj, że ich łączenie bywa trudne.

Ta książka pomoże Ci podejmować decyzje związane z wyborem technologii i ich łączeniem. Jak się przekonasz, żadne podejście nie jest fundamentalnie lepsze od innych. Każde ma określone plusy i minusy. Dzięki tej książce nauczysz się zadawać właściwe pytania w trakcie oceny i porównywania systemów zarządzania danymi, aby móc wybrać rozwiązanie najlepiej odpowiadające potrzebom budowanej aplikacji.

Omówienie zaczniemy od przyjrzenia się typowym sposobom korzystania z danych w dzisiejszych organizacjach. Wiele z omawianych tu koncepcji wywodzi się z **oprogramowania korporacyjnego**, związanego z potrzebami i praktykami inżynieryjnymi dużych organizacji, takich jak wielkie korporacje i instytucje państwowe, ponieważ historycznie tylko duże organizacje dysponowały wolumenami danych wymagającymi zaawansowanych rozwiązań technicznych. Jeśli posiadasz niewiele danych, możesz je przechowywać nawet w arkuszu kalkulacyjnym! Jednak ostatnio również mniejsze firmy i startupy zaczęły zarządzać dużymi wolumenami danych i budować intensywnie przetwarzające je systemy.

Jednym z głównych wyzwań związanych z systemami danych jest to, że różne osoby potrzebują danych do wykonywania zupełnie innych zadań. Jeśli pracujesz w firmie, Ty i Twój zespół macie określone priorytety, podczas gdy inna grupa może mieć całkiem inne cele, mimo że wszyscy korzystają z tego samego zbioru danych! Ponadto cele te mogą nie być jasno sformułowane, co prowadzi do nieporozumień i sporów o właściwe podejście.

Aby pomóc Ci zrozumieć dostępne możliwości, w tym rozdziale porównujemy kilka zróżnicowanych koncepcji i analizujemy związane z nimi kompromisy. Oto omawiane zagadnienia:

- różnice między systemami transakcyjnymi a analitycznymi (podrozdział „Systemy transakcyjne a systemy analityczne”);
- wady i zalety usług chmurowych oraz systemów hostowanych samodzielnie (podrozdział „Chmura a hosting własny”);
- przechodzenie z systemów jednowęzłowych na systemy rozproszone (podrozdział „Systemy rozproszone a systemy jednowęzłowe”);

Kup książkę Poleć książkę

- zachowanie równowagi między potrzebami biznesu a prawami użytkownika (podrozdział „Systemy danych, prawo i społeczeństwo”).

Ponadto w tym rozdziale przedstawiamy terminologię, która będzie potrzebna w dalszej części książki.

Terminologia: frontendy i backendy

Znaczna część zagadnień omawianych w tej książce dotyczy **rozwoju backendu**. Warto wyjaśnić ten termin. W aplikacjach internetowych kod działający po stronie klienta (czyli w przeglądarce internetowej) nazywamy **frontendem**, natomiast kod po stronie serwera, obsługujący żądania użytkowników, to właśnie **backend**. Aplikacje mobilne są podobne do frontendów w tym sensie, że zapewniają interfejs użytkownika, który często komunikuje się przez internet z działającym po stronie serwera backendem. Frontendy czasami zarządzają danymi lokalnie na urządzeniu użytkownika [2], jednak największe wyzwania związane z infrastrukturą danych leżą zazwyczaj po stronie backendu. Dzieje się tak, ponieważ frontend musi obsługiwać dane tylko jednego użytkownika, podczas gdy backend zarządza danymi na potrzeby *wszystkich* użytkowników.

Usługa backendowa jest przeważnie dostępna przez protokół HTTP (czasem WebSocket) i składa się zazwyczaj z kodu aplikacji, który wczytuje i zapisuje dane z użyciem jednej lub kilku baz, a niekiedy komunikuje się z dodatkowymi systemami zarządzania danymi, takimi jak pamięć podręczna czy kolejki wiadomości (które to mechanizmy można zbiorczo nazwać **infrastrukturą danych**). Kod aplikacji często ma charakter *bezstanowy*, co oznacza, że po zakończeniu obsługi jednego żądania HTTP kod zapomina wszystko na ten temat. Wszelkie informacje, które trzeba zachować między kolejnymi żądaniami, należy przechowywać albo po stronie klienta, albo w infrastrukturze danych po stronie serwera.

Systemy transakcyjne a systemy analityczne

Jeśli pracujesz z systemami danych w przedsiębiorstwie, prawdopodobnie napotkasz kilka różnych grup osób zajmujących się danymi. Pierwszą z nich są **inżynierowie backendów**, którzy tworzą usługi obsługujące żądania odczytu i aktualizacji danych. Usługi te często obsługują użytkowników zewnętrznych, bezpośrednio lub pośrednio z użyciem innych serwisów (zobacz punkt „Mikrousługi i architektura bezserwerowa”). Czasami usługi są przeznaczone do użytku wewnętrznego w innych częściach organizacji.

Oprócz zespołów zarządzających usługami backendowymi dostępu do danych organizacji zazwyczaj potrzebują także dwie inne grupy osób: **analitycy biznesowi**, generujący raporty o działalności organizacji, aby pomóc kierownictwu podejmować lepsze decyzje (**analityka biznesowa**), i **danologowie**, którzy wyciągają nowe wnioski na podstawie danych albo opracowują dla użytkowników funkcje produktowe oparte na analizie danych i uczeniu maszynowym bądź sztucznej inteligencji (na przykład rekomendacje „klienci, którzy kupili X, kupili też Y” w sklepie internetowym, funkcje związane z analityką predykcyjną, w tym ocena ryzyka lub filtrowanie spamu, czy też ranking wyników wyszukiwania).

Kup książkę Poleć książkę

Chociaż analitycy biznesowi i danologowie używają różnych narzędzi i pracują w odmienny sposób, mają pewne cechy wspólne: obie grupy zajmują się *analityką*, co oznacza, że badają dane wygenerowane przez użytkowników i usługi backendowe, ale zwykle tych danych nie modyfikują (z wyjątkiem ewentualnego poprawiania błędów). Mogą jednak tworzyć pochodne zbiory danych, w których oryginalne dane zostały w jakiś sposób przetworzone. Prowadzi to do podziału na dwa typy systemów, którego będziemy używać w całej książce:

- **Systemy transakcyjne** składają się z usług backendowych i infrastruktury danych, gdzie są one generowane (na przykład w procesie obsługi użytkowników zewnętrznych). W takich systemach kod aplikacji zarówno odczytuje, jak i modyfikuje dane w bazach w zależności od operacji wykonywanych przez użytkowników.
- **Systemy analityczne** służą potrzebom analityków biznesowych i danologów. Zawierają kopię dostępnych tylko do odczytu danych z systemów transakcyjnych i są zoptymalizowane pod kątem przetwarzania danych na potrzeby analityki.

Jak wyjaśniamy w następnym punkcie, systemy transakcyjne i analityczne są często utrzymywane oddzielnie. Wynika to z dobrych powodów. W miarę dojrzewania takich systemów pojawiły się dwie nowe wyspecjalizowane role: inżynierowie danych i inżynierowie analityki. **Inżynierowie danych** to osoby, które wiedzą, jak integrować systemy transakcyjne z analitycznymi. Grupa ta odpowiada za infrastrukturę danych organizacji w szerszym zakresie [3]. Natomiast **inżynierowie analityki** modelują i przekształcają dane, aby były bardziej użyteczne dla analityków biznesowych i danologów w organizacji [4].

Wielu inżynierów specjalizuje się albo w systemach transakcyjnych, albo w systemach analitycznych. Jednak w tej książce uwzględniamy zarówno transakcyjne, jak i analityczne systemy danych, ponieważ oba odgrywają ważną rolę w cyklu życia danych w organizacji. Szczegółowo badamy infrastrukturę danych wykorzystywaną do udostępniania usług zarówno użytkownikom wewnętrznym, jak i zewnętrznym, co pomoże Ci lepiej współpracować z osobami zajmującymi się innym obszarem niż Twój.

Charakterystyka przetwarzania transakcji i analityki

W początkach rozwoju branży przetwarzania danych biznesowych zapis w bazie zazwyczaj wiązał się z przeprowadzeniem **transakcji handlowej**: sprzedażą, złożeniem zamówienia u dostawcy, wypłatą pensji pracownikowi itp. Choć później bazy danych zaczęto wykorzystywać w obszarach niezwiązanych z przepływem pieniędzy, termin **transakcja** nadal jest używany. Obecnie opisuje on grupę odczytów i zapisów tworzących logiczną całość.

W rozdziale 8. szczegółowo wyjaśniamy, co rozumiemy przez transakcję. W niniejszym rozdziale termin ten jest używany na bardziej ogólnym poziomie i oznacza odczyty i zapisy wykonywane z niewielkim opóźnieniem.

Choć bazy danych zaczęto wykorzystywać do przechowywania różnorodnych danych — postów w mediach społecznościowych, ruchów w grach, kontaktów w książce adresowej itd. — podstawowy wzorzec dostępu wciąż wygląda podobnie jak dla przetwarzania transakcji biznesowych.

Kup książkę Poleć książkę

System transakcyjny przeważnie wyszukuje niewielką liczbę rekordów według klucza (nazywa się to **zapytaniem punktowym**). Rekordy są wstawiane, aktualizowane lub usuwane na podstawie danych wprowadzonych przez użytkownika. Ponieważ to podejście jest interaktywne, ten wzorzec dostępu zaczęto nazywać modelem **OLTP** (ang. *online transaction processing*).

Jednak coraz częściej bazy danych zaczęto wykorzystywać również do analityki, która charakteryzuje się zupełnie innymi wzorcami dostępu niż w modelu OLTP. Zapytanie analityczne zazwyczaj wymaga przeszukania ogromnej liczby rekordów i obliczenia statystyk zbiorczych (takich jak liczba wystąpień, suma czy średnia), zamiast zwracać użytkownikowi poszczególne rekordy. Na przykład analityk biznesowy pracujący dla sieci supermarketów może chcieć uzyskać odpowiedzi na następujące pytania analityczne:

- Ile wyniósł łączny przychód w każdym z naszych sklepów w styczniu?
- O ile więcej niż zwykle sprzedaliśmy bananów w trakcie najnowszej promocji?
- Jaka marka posiłków dla niemowląt jest najczęściej kupowana razem z pieluchami marki X?

Raporty powstające na podstawie tego typu zapytań są istotne w analityce biznesowej i pomagają kierownictwu w podejmowaniu decyzji. Aby odróżnić ten wzorzec używania baz od przetwarzania transakcji, nazwano go **OLAP** (ang. *online analytic processing*, czyli przetwarzanie analityczne w czasie rzeczywistym) [5]. Różnice między podejściami OLTP i OLAP nie zawsze są wyraźne, jednak typowe cechy obu modeli zostały wymienione w tabeli 1.1.

Tabela 1.1. Porównanie cech przetwarzania transakcji i systemów analitycznych

Cecha	Systemy OLTP	Systemy OLAP
Podstawowy wzorzec odczytu	Zapytania punktowe (pobieranie pojedynczych rekordów na podstawie klucza)	Agregowanie z użyciem dużej liczby rekordów
Podstawowy wzorzec zapisu	Tworzenie, aktualizowanie i usuwanie pojedynczych rekordów	Import masowy (ETL) lub strumieniowanie zdarzeń
Przykład użytkowania przez człowieka	Użytkownik końcowy aplikacji internetowej lub mobilnej	Analityk wewnętrzny (do wspomagania podejmowania decyzji)
Przykład użytkowania przez maszynę	Sprawdzanie uprawnień	Wykrywanie wzorców wskazujących na oszustwa lub nadużycia
Typ zapytań	Stały zestaw zapytań zdefiniowanych w aplikacji	Analityk może uruchamiać dowolne zapytania
Profil zapytań	Wiele prostych zapytań	Niewielka liczba zapytań, ale wszystkie są rozbudowane
Dane reprezentują	Najnowszy stan danych (aktualny moment w czasie)	Historię zdarzeń zachodzących w czasie
Wielkość zbioru danych	Od gigabajtów do terabajtów	Od terabajtów do petabajtów

Znaczenie słowa *online* w nazwie OLAP jest niejasne. Prawdopodobnie chodzi o to, że zapytania są przeznaczone nie tylko dla wcześniej zdefiniowanych raportów, lecz analitycy używają systemu OLAP interaktywnie na potrzeby zapytań eksploracyjnych.

Kup książkę Poleć książkę

W systemach transakcyjnych użytkownicy zazwyczaj nie mają możliwości tworzenia własnych zapytań SQL-owych i uruchamiania ich w bazie danych, ponieważ mogłoby to pozwolić na odczyt lub modyfikację danych, do których dana osoba nie ma uprawnień. Ponadto użytkownicy mogliby pisać zapytania kosztowne obliczeniowo, wpływające negatywnie na wydajność bazy z perspektywy innych osób. Z tych powodów systemy OLTP najczęściej wykonują ustalony zestaw zapytań wbudowanych w kod aplikacji, a niestandardowe jednorazowe zapytania stosuje się tylko okazjonalnie w celach konserwacyjnych lub diagnostycznych. Z kolei analityczne bazy danych przeważnie dają użytkownikom swobodę ręcznego pisania dowolnych zapytań SQL-owych lub automatycznego generowania ich za pomocą narzędzi do wizualizacji danych bądź paneli kontrolnych (dotyczy to na przykład aplikacji Tableau, Looker czy Microsoft Power BI).

Istnieje również kategoria systemów zaprojektowanych do wykonywania zadań analitycznych (czyli do obsługi zapytań agregujących wiele rekordów), które są jednak wbudowane w produkty przeznaczone dla użytkowników końcowych. Ta kategoria to narzędzia do **analityki produktowej** lub **analityki w czasie rzeczywistym**. Systemy zaprojektowane do tego typu zastosowań to między innymi Pinot, Druid i ClickHouse [6]. Takie systemy pobierają dane w czasie rzeczywistym i są zoptymalizowane w taki sposób, by szybko generowały odpowiedzi na zapytania. W przeciwieństwie do takich rozwiązań tradycyjne systemy OLAP zazwyczaj pobierają dane w partiach i są zoptymalizowane pod kątem wysokiej przepustowości przetwarzania zapytań, a nie niskich opóźnień.

Hurtownie danych

Początkowo te same bazy danych służyły zarówno do przetwarzania transakcji, jak i do obsługi zapytań analitycznych. SQL okazał się w tym zakresie dość elastyczny, ponieważ sprawdza się dobrze w obu typach zapytań. Niemniej jednak pod koniec lat 80. i na początku lat 90. firmy zaczęły odchodzić od wykorzystywania systemów OLTP do celów analitycznych. Do analiz zaczęto stosować odrębne systemy bazodanowe, nazywane hurtowniami danych.

Duża firma może używać dziesiątek, a nawet setek różnych systemów przetwarzania transakcji: systemów obsługujących udostępnianą użytkownikom witrynę, kontrolujących systemy w punktach sprzedaży (kasy) w fizycznych sklepach, śledzących stan magazynu, planujących trasy pojazdów, zarządzających dostawami i pracownikami itd. Każdy z tych systemów jest złożony i wymaga zespołu osób do konserwacji. Dlatego takie systemy zwykle działają niezależnie od siebie.

Bezpośrednie kierowanie zapytań do systemów OLTP przez analityków biznesowych i danologów jest zazwyczaj niepożądane. Wynika to z kilku przyczyn:

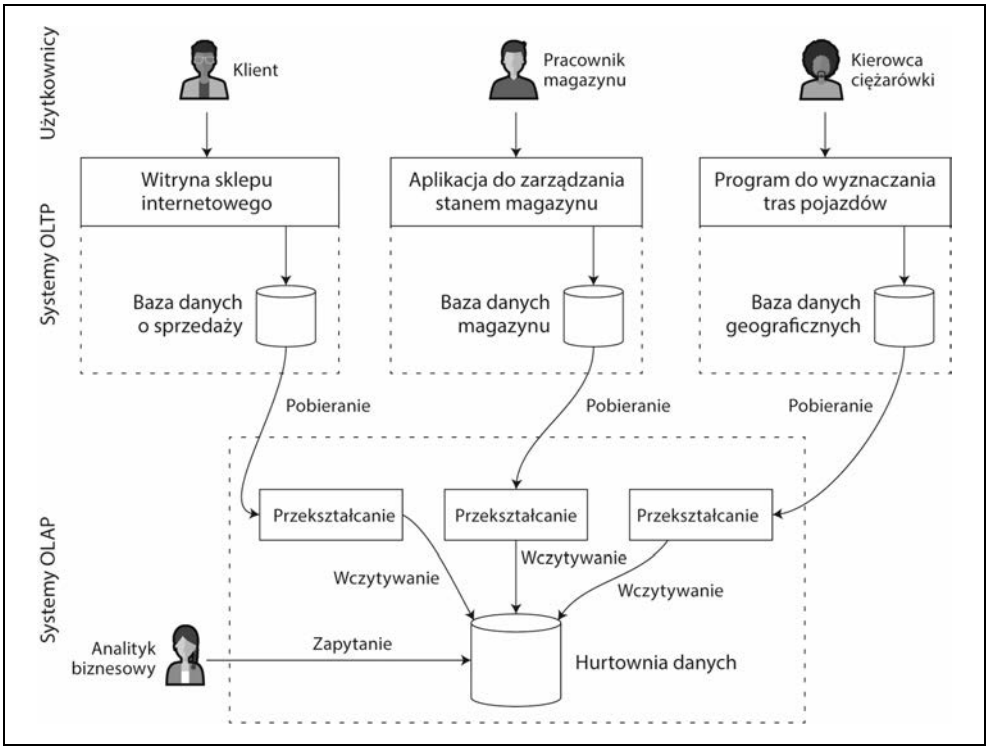
- interesujące dane mogą być rozproszone w wielu systemach transakcyjnych, co utrudnia łączenie tych zbiorów danych w jednym zapytaniu (jest to problem **silosów danych**);
- schematy i układy danych odpowiednie dla OLTP są gorzej przystosowane do analityki (zobacz punkt „Gwiazdy i płatki śniegu — schematy używane w analityce” w rozdziale 3.);
- zapytania analityczne mogą być dość kosztowne, a ich uruchamianie w bazie OLTP wpływa na wydajność systemu z perspektywy innych użytkowników;

Kup książkę Poleć książkę

- systemy OLTP mogą znajdować się w oddzielnej sieci, do której użytkownicy nie mają bezpośredniego dostępu ze względów bezpieczeństwa lub zgodności z regulacjami.

Natomiast **hurtownia danych** to oddzielna baza, do której analitycy mogą swobodnie kierować zapytania bez wpływu na działanie systemów OLTP [7]. W rozdziale 4. wyjaśniamy, że hurtownie danych często przechowują dane w sposób znacząco odmienny niż bazy OLTP. Hurtownie danych optymalizuje się pod kątem typowych zapytań analitycznych.

Hurtownia danych obejmuje przeznaczoną tylko do odczytu kopię danych z różnych systemów OLTP firmy. Dane są pobierane z baz OLTP (albo za pomocą okresowego zrzutu danych, albo w ramach ciągłego strumienia aktualizacji), przekształcane na format ułatwiający analizy, oczyszczane, a następnie wczytywane do hurtowni danych. Proces przenoszenia danych do hurtowni to **ETL** (ang. *Extract-Transform-Load*, czyli pobieranie, przekształcanie, wczytywanie). Przedstawia go rysunek 1.1. Czasem kolejność *przekształcania* i *wczytywania* jest zmieniana (transformacje odbywają się wtedy w hurtowni już po wczytaniu danych). Wtedy proces nazywany jest **ELT**.



Rysunek 1.1. Uproszczony zarys procesu ETL wykonywanego na potrzeby hurtowni danych

W niektórych przypadkach źródłami danych dla procesów ETL są zewnętrzne produkty SaaS, na przykład systemy do zarządzania relacjami z klientami (CRM), e-mail marketingu czy obsługi płatności kartami. W takich sytuacjach oryginalna baza danych nie jest bezpośrednio dostępna. Można z niej korzystać wyłącznie za pośrednictwem API dostawcy oprogramowania.

Kup książkę Poleć książkę

Przeniesienie danych z zewnętrznych systemów do własnej hurtowni danych umożliwia przeprowadzanie analiz, które nie są możliwe za pośrednictwem API produktów SaaS. Procesy ETL z użyciem API produktów SaaS są często obsługiwane przez wyspecjalizowane usługi do integracji danych, takie jak Fivetran, Singer czy AirByte.

Niektóre systemy bazodanowe oferują **hybrydowe przetwarzanie transakcyjno-analityczne** (ang. *hybrid transactional/analytic processing* — HTAP), co ma umożliwić obsługę OLTP i analityki w jednym systemie, bez konieczności stosowania procesów ETL do przenoszenia danych między systemami [8, 9]. Jednak wiele rozwiązań HTAP udostępnia wspólny interfejs, ale wewnętrznie obejmuje system OLTP połączony z oddzielnym systemem analitycznym, dlatego rozróżnienie między tymi wewnętrznymi komponentami pozostaje istotne dla zrozumienia działania rozwiązań HTAP.

Co więcej, mimo dostępności rozwiązań HTAP powszechną praktyką jest rozdzielanie systemów transakcyjnych i analitycznych ze względu na związane z nimi odmienne cele i wymagania. Przede wszystkim do dobrych praktyk należy, by każdy system transakcyjny miał własną bazę danych (zobacz punkt „Mikrousługi i architektura bezserwerowa" w tym rozdziale), co prowadzi do tworzenia setek oddzielnych transakcyjnych baz danych. Ale większość przedsiębiorstw posiada jedną hurtownię, dzięki czemu analitycy biznesowi mogą łączyć dane z wielu systemów transakcyjnych w jednym zapytaniu.

Rozwiązania HTAP nie zastępują zatem hurtowni danych. Są natomiast przydatne wtedy, gdy ta sama aplikacja musi zarówno wykonywać zapytania analityczne skanujące dużą liczbę wierszy, jak i wczytywać oraz aktualizować pojedyncze rekordy z niskim opóźnieniem. Przykładem takiego zadania może być wykrywanie oszustw [10].

Rozdzielanie systemów transakcyjnych i analitycznych jest częścią szerszego trendu: gdy procesy stają się bardziej wymagające, systemy muszą być coraz bardziej wyspecjalizowane i lepiej zoptymalizowane na potrzeby konkretnych zadań. Systemy ogólnego przeznaczenia radzą sobie bezproblemowo z małymi wolumenami danych, ale im większa skala, tym bardziej systemy stają się wyspecjalizowane [11].

Od hurtowni danych do jeziora danych

W hurtowniach danych często wykorzystuje się model *relacyjny* z zapytaniami uruchamianymi w SQL-u (zobacz rozdział 3.), czasem z użyciem specjalistycznego oprogramowania do analityki biznesowej. Model ten sprawdza się dobrze w zapytaniach, jakie muszą wykonywać analitycy biznesowi, ale jest mniej odpowiedni dla danologów, którzy mogą potrzebować wykonywać takie zadania jak:

- Przekształcanie danych na postać odpowiednią do trenowania modeli w uczeniu maszynowym, co często wymaga zamiany wierszy i kolumn tabeli bazy danych na wektor lub macierz wartości liczbowych zwanych *cechami*. Proces przeprowadzania takich przekształceń w sposób maksymalizujący skuteczność wytrenowanego modelu nazywa się **inżynierią cech**. Często niezbędny jest do tego niestandardowy kod, który trudno jest napisać za pomocą SQL-a.

Kup książkę Poleć książkę

- Pobieranie danych tekstowych (na przykład recenzji produktu) i wykorzystywanie technik przetwarzania języka naturalnego do wyodrębniania z nich ustrukturyzowanych informacji (takich jak nastawienie autora lub tematy, które porusza). Czasem konieczne jest też wyodrębnianie ustrukturyzowanych informacji ze zdjęć za pomocą technik widzenia komputerowego.

Chociaż podejmowano próby dodawania do SQL-owego modelu danych operatorów potrzebnych w uczeniu maszynowym [12] oraz budowania wydajnych relacyjnych systemów uczenia maszynowego [13], wielu danologów woli nie pracować na poziomie relacyjnych baz danych, takich jak hurtownie danych, i preferuje biblioteki Pythona przeznaczone do analizy danych, takie jak *pandas* i *scikit-learn*, języki do analizy statystycznej, na przykład R, oraz rozproszone silniki przetwarzania danych, takie jak Spark [14]. Te rozwiązania omawiamy szerzej w podrozdziale „Ramki danych, macierze i tablice" w rozdziale 3.

Z wymienionych przyczyn organizacje stają przed koniecznością udostępniania danych w formie odpowiedniej dla danologów. Rozwiązaniem jest **jezioro danych**, czyli scentralizowane repozytorium przechowujące kopie wszelkich danych, które mogą być przydatne w analizach. Dane te są pobierane z systemów transakcyjnych za pomocą procesów ETL. Różnica w porównaniu z hurtownią danych polega na tym, że jezioro danych zawiera dowolne pliki, bez narzucania konkretnego formatu, modelu danych czy schematu [15]. Pliki w jeziorze danych mogą być kolekcjami rekordów bazodanowych zakodowanych w formatach takich jak Avro czy Parquet (zobacz rozdział 5.), ale równie dobrze mogą zawierać tekst, obrazy, filmy, odczyty z czujników, macierze rzadkie, wektory cech, sekwencje genomowe lub dane dowolnego innego rodzaju [16]. Takie rozwiązanie nie tylko daje większą elastyczność, ale też często okazuje się tańsze niż przechowywanie danych w modelu relacyjnym. Dzieje się tak, ponieważ w jeziorach danych można stosować ekonomiczne sposoby przechowywania plików, takie jak pamięć obiektowa (zobacz punkt „Architektura systemów natywnych dla chmury" w tym rozdziale).

Procesy ETL zostały uogólnione do postaci *potoków danych*, a w niektórych scenariuszach jezioro danych jest etapem pośrednim między systemami transakcyjnymi a hurtownią danych. Jezioro danych zawiera dane w nieprzetworzonej postaci, tak jak zostały wygenerowane w systemach transakcyjnych, bez przekształcania ich na schemat używany w relacyjnej hurtowni danych. Takie podejście ma tę zaletę, że każdy użytkownik może przekształcić surowe dane do formy najlepiej odpowiadającej jego potrzebom. Zostało to nazwane *zasadą sushi*: „surowe dane są lepsze" [17].

Poza jeziorem danych

Wraz z dojrzewaniem praktyk analitycznych organizacje coraz większą uwagę poświęcają zarządzaniu systemami analitycznymi i potokami danych oraz ich obsłudze, co zostało opisane między innymi w manifeście DataOps [18]. Związane są z tym kwestie nadzoru, prywatności oraz zgodności z regulacjami takimi jak RODO i CCPA, co omawiamy w podrozdziale „Systemy danych, prawo i społeczeństwo" oraz w rozdziale 14.

Kup książkę Poleć książkę

Ponadto dane analityczne są coraz częściej udostępniane nie tylko w postaci plików i tabel relacyjnych, ale również jako strumienie zdarzeń (zobacz rozdział 12.). Analizy oparte na plikach można przeprowadzać okresowo (na przykład codziennie), aby reagować na zmiany w danych, natomiast przetwarzanie strumieniowe pozwala systemom analitycznym reagować na zdarzenia znacznie szybciej, w czasie mierzonym w sekundach. W zależności od aplikacji i jej wrażliwości na czas podejście wykorzystujące przetwarzanie strumieniowe może okazać się wartościowe. Dotyczy to na przykład identyfikowania i blokowania potencjalnie nieuczciwych lub szkodliwych działań.

W niektórych scenariuszach wyniki wygenerowane w systemach analitycznych są udostępniane systemom transakcyjnym (proces ten to **odwrotny ETL**; ang. *reverse ETL* [19]). Na przykład model uczenia maszynowego wytrenowany na danych w systemie analitycznym może zostać wdrożony w środowisku produkcyjnym, aby generować dla użytkowników końcowych rekomendacje typu „klienci, którzy kupili X, kupili również Y". Modele uczenia maszynowego można wdrażać w systemach operacyjnych za pomocą wyspecjalizowanych narzędzi, takich jak TFX, Kubeflow czy MLflow.

Systemy zapisu a systemy danych pochodnych

Powiązane z podziałem na systemy operacyjne i analityczne jest rozróżnienie na **systemy zapisu** i **systemy danych pochodnych**. Te terminy są przydatne, ponieważ pomagają zrozumieć przepływ danych w systemie:

Systemy zapisu

System zapisu, nazywany też *źródłem faktów*, przechowuje wiarygodną, *kanoniczną* wersję danych. Gdy pojawiają się nowe dane (na przykład dane wejściowe od użytkownika), najpierw są zapisywane właśnie tu. Każdy fakt jest reprezentowany tylko raz (reprezentacja jest zwykle *znormalizowana*; zobacz punkt „Normalizacja, denormalizacja i złączenia" w rozdziale 3.). Jeśli występuje rozbieżność między innym systemem a systemem zapisu, z definicji poprawna jest wartość z systemu zapisu.

Systemy danych pochodnych

Informacje w systemie danych pochodnych są wynikiem pobrania istniejących danych z innego systemu i przekształcenia lub przetworzenia ich w jakiś sposób. Jeśli utracisz dane pochodne, możesz je odtworzyć na podstawie danych źródłowych. Klasycznym przykładem jest pamięć podręczna. Dane mogą być udostępniane z pamięci podręcznej, jeśli są tam dostępne. Jeżeli jednak taka pamięć nie zawiera potrzebnych danych, można wykorzystać używaną bazę. Do tej kategorii należą wartości zdenormalizowane, indeksy, widoki zmaterializowane, przekształcone reprezentacje danych i modele wyuczone na podstawie zbioru danych.

Technicznie rzecz biorąc, dane pochodne są *nadmiarowe* — w tym sensie, że powielają istniejące informacje. Jednak często okazują się niezbędne do uzyskania wysokiej wydajności zapytań wymagających odczytu. Na podstawie jednego źródła można wygenerować kilka różnych zbiorów danych, co pozwala przyjrzeć się danym z paru punktów widzenia.

Kup książkę Poleć książkę

Systemy analityczne są zazwyczaj systemami danych pochodnych, ponieważ korzystają z danych powstałych w innym miejscu. Usługi operacyjne mogą zawierać połączenie systemów zapisu i systemów danych pochodnych. Systemy zapisu to główne bazy danych, w których dane są umieszczane w pierwszej kolejności, natomiast systemy danych pochodnych to indeksy i pamięć podręczna przyspieszające typowe operacje odczytu, szczególnie dla zapytań, których system zapisu nie jest w stanie efektywnie obsłużyć.

Większość baz danych, systemów składowania danych i języków zapytań nie jest z natury systemem zapisu lub systemem danych pochodnych. Baza danych to tylko narzędzie. To, jak będziesz jej używał, zależy od Ciebie. Rozróżnienie na systemy zapisu i systemy danych pochodnych zależy nie od narzędzia, ale od sposobu wykorzystania go w aplikacji. Dzięki jednoznacznemu określeniu, które dane są pochodne od poszczególnych innych danych, można jasno opisać skomplikowaną architekturę systemu.

Gdy dane w jednym systemie są tworzone na podstawie informacji z innego miejsca, potrzebny jest proces aktualizacji danych pochodnych w momencie modyfikacji oryginału w systemie zapisu. Niestety wiele baz danych jest projektowanych przy założeniu, że aplikacja będzie korzystać tylko z tej jednej bazy, przez co trudno jest zintegrować wiele systemów w celu przekazywania aktualizacji. W rozdziale 11. omawiamy *integrację danych* z wykorzystaniem potoków, które pozwalają łączyć wiele systemów w celu wykonywania zadań niemożliwych do zrealizowania w jednym narzędziu.

To kończy porównanie analityki i przetwarzania transakcyjnego. W następnym podrozdziale omawiamy kolejny kompromis. Zapewne wielokrotnie zdarzyło Ci się słuchać dyskusji na ten temat.

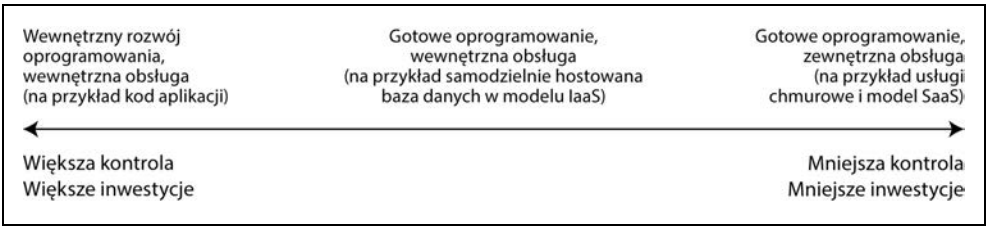
Chmura a hosting własny

Przy każdym zadaniu, które organizacja musi wykonać, jedno z pierwszych pytań brzmi: czy lepiej zrobić to samodzielnie, czy zlecić na zewnątrz? Budować własne rozwiązanie czy kupić gotowe?

Ostatecznie zależy to od priorytetów biznesowych. Zgodnie z powszechnie przyjętą wiedzą z obszaru zarządzania aspekty związane z kluczowymi kompetencjami lub przewagą konkurencyjną organizacji powinny być rozwijane wewnętrznie, natomiast zadania poboczne, rutynowe lub typowe należy zlecać dostawcom zewnętrznym [20]. Skrajnym przykładem jest to, że większość firm nie produkuje własnych procesorów, ponieważ taniej jest kupić je od producentów półprzewodników.

Jeśli chodzi o oprogramowanie, trzeba podjąć dwie ważne decyzje: kto ma je rozwijać i kto ma je wdrażać. Istnieje całe spektrum możliwości związanych z różnym stopniem zlecania prac firmom zewnętrznym, co ilustruje rysunek 1.2. Na jednym końcu znajduje się oprogramowanie dedykowane, rozwijane i uruchamiane we własnym zakresie. Na drugim końcu są powszechnie używane usługi chmurowe lub produkty typu SaaS, budowane i obsługiwane przez zewnętrznego dostawcę. Użytkownicy korzystają z nich jedynie przez interfejs internetowy lub API.

Kup książkę Poleć książkę



Rysunek 1.2. Spektrum typów oprogramowania i sposobów jego eksploatacji

Rozwiązaniem pośrednim jest gotowe oprogramowanie (otwartoźródłowe lub komercyjne), które firma *hostuje samodzielnie*. Wdraża je więc wewnętrznie, na przykład pobiera system MySQL i instaluje go na kontrolowanym przez siebie serwerze. Może to być jej własny sprzęt (często nazywany **infrastrukturą lokalną**, nawet jeśli serwer faktycznie znajduje się w wynajętej szafce w centrum danych, a nie dosłownie w siedzibie firmy) lub maszyna wirtualna w chmurze (**model IaaS**, ang. *Infrastructure as a Service*). Na tej skali istnieje jeszcze więcej punktów pośrednich. Można na przykład uruchomić zmodyfikowaną wersję oprogramowania otwartoźródłowego.

Niezależnie od tego spektrum ważna jest również kwestia *sposobu wdrażania* usług — zarówno w chmurze, jak i lokalnie. Należy na przykład uwzględnić, czy firma będzie korzystać z platformy do orkiestracji, takiej jak Kubernetes. Jednak omawianie wyboru narzędzi do wdrażania wykracza poza zakres tej książki, ponieważ inne czynniki mają większy wpływ na architekturę systemów danych.

Wady i zalety usług chmurowych

Korzystanie z usługi chmurowej zamiast samodzielnego uruchamiania podobnego oprogramowania oznacza w zasadzie zlecenie obsługi tego oprogramowania dostawcy chmury. Istnieją dobre argumenty zarówno za usługami chmurowymi, jak i przeciw nim. Dostawcy chmury twierdzą, że korzystanie z ich usług oszczędza czas i pieniądze oraz pozwala działać szybciej w porównaniu z budowaniem własnej infrastruktury.

To, czy usługa chmurowa jest faktycznie tańsza i łatwiejsza w użyciu niż samodzielne hostowanie, zależy w dużej mierze od Twoich umiejętności i obciążenia systemów. Jeśli masz już doświadczenie w konfigurowaniu i obsłudze potrzebnych systemów, a obciążenie jest dość przewidywalne (czyli liczba potrzebnych maszyn nie zmienia się w znacznym stopniu), często taniej jest kupić własne maszyny i samodzielnie uruchomić na nich oprogramowanie [21, 22].

Jeżeli jednak nie wiesz, jak wdrożyć i obsługiwać potrzebny system, to zastosowanie usługi chmurowej jest często łatwiejsze i szybsze niż samodzielna nauka zarządzania odpowiednim narzędziem. Jeśli firma musi zatrudnić i przeszkolić pracowników w celu utrzymania i obsługi systemu, może to okazać się bardzo kosztowne. Gdy korzystasz z chmury, nadal potrzebujesz zespołu operacyjnego (zobacz punkt „Operacje w erze chmury" w tym rozdziale), ale zlecenie podstawowych aspektów administracji systemem na zewnątrz pozwala uwolnić Twój zespół od dodatkowych obowiązków, dzięki czemu pracownicy mogą skupić się na sprawach wyższego poziomu.

Kup książkę Poleć książkę

Kiedy zlecasz obsługę systemu firmie specjalizującej się w takich zadaniach, może to prowadzić do wyższego poziomu usługi, ponieważ dostawca zdobywa wiedzę operacyjną dzięki pracy dla wielu klientów. Z kolei jeśli samodzielnie zarządzasz usługą, możesz ją skonfigurować i dostroić tak, by działała dobrze przy Twoim konkretnym obciążeniu. Jest mało prawdopodobne, że operatorzy usługi chmurowej będą skłonni dokonywać takich zmian na Twoje życzenie.

Usługi chmurowe są szczególnie wartościowe, gdy obciążenie systemów znacznie zmienia się w czasie. Jeśli konfigurujesz maszyny w taki sposób, by radziły sobie z obciążeniem szczytowym, to zasoby obliczeniowe przez większość czasu pozostają niewykorzystane, a system staje się mniej opłacalny. W takim scenariuszu usługi chmurowe mają tę zaletę, że ułatwiają skalowanie zasobów obliczeniowych w górę lub w dół w reakcji na zmiany zapotrzebowania.

Na przykład systemy analityczne często mają wysoce zmienne obciążenie. By szybko obsłużyć rozbudowane zapytanie analityczne, potrzeba wielu równoległych zasobów obliczeniowych. Jednak po zakończeniu udzielania odpowiedzi zasoby pozostają bezczynne do momentu, gdy użytkownik uruchomi kolejne zapytanie. Predefiniowane zapytania (używane na przykład na potrzeby codziennych raportów) można umieszczać w kolejce i wykonywać zgodnie z harmonogramem, aby wyrównać obciążenie. Ale jeśli chcesz szybko uzyskiwać odpowiedzi na zapytania interaktywne, to obciążenie z natury będzie zmienne. Jeżeli używany zbiór danych jest tak duży, że szybka obsługa zapytań wymaga znacznych zasobów obliczeniowych, korzystanie z chmury pozwala zaoszczędzić pieniądze, ponieważ można zwrócić niewykorzystane zasoby dostawcy, zamiast pozostawiać je bezczynne. Przy mniejszych zbiorach danych ta różnica jest mniej istotna.

Największą wadą usługi chmurowej jest brak kontroli:

- Jeśli brakuje Ci potrzebnej funkcji, możesz jedynie grzecznie poprosić dostawcę o jej dodanie. Zazwyczaj nie możesz zaimplementować jej samodzielnie.
- Jeżeli usługa przestanie działać, możesz tylko czekać na jej przywrócenie.
- Jeśli korzystasz z usługi w sposób, który wywołuje błąd lub powoduje problemy z wydajnością, trudno będzie Ci zdiagnozować problem. W oprogramowaniu, które uruchamiasz samodzielnie, możesz uzyskać wskaźniki wydajności i informacje diagnostyczne z systemu operacyjnego, co pomaga zrozumieć jego zachowanie. Możesz też przejrzeć dzienniki serwera. W usłudze hostowanej przez dostawcę przeważnie nie masz dostępu do tego rodzaju wewnętrznych danych.
- Co więcej, jeśli usługa zostanie zamknięta, stanie się zbyt droga lub dostawca zdecyduje się zmienić produkt w sposób, który nam nie odpowiada, jesteśmy całkowicie od niego zależni. Kontynuowanie pracy z użyciem starej wersji oprogramowania zazwyczaj nie jest możliwe, dlatego firma jest zmuszona do wyboru innej usługi i przeprowadzenia migracji [23]. Ryzyko to jest mniejsze, gdy istnieją inne usługi oferujące kompatybilne API, jednak wiele usług chmurowych nie ma standardowych interfejsów, co podnosi koszty przejścia i sprawia, że uzależnienie od dostawcy staje się realnym problemem.
- Jeśli dostawca chmury działa w innym państwie, które znajdzie się w stanie konfliktu z krajem klienta, powstaje ryzyko braku dostępu do usługi z powodu nałożonych sankcji.

Kup książkę Poleć książkę

- Dostawca chmury musi być godny zaufania w kwestii bezpieczeństwa danych, przez co może być trudniej spełnić wymogi przepisów z zakresu prywatności i bezpieczeństwa.

Pomimo wszystkich tych zagrożeń coraz więcej organizacji decyduje się budować nowe aplikacje z wykorzystaniem usług chmurowych lub stosuje podejście hybrydowe, w którym w chmurze działają wybrane elementy systemu. Usługi chmurowe nie zastąpią jednak całkowicie wewnętrznych systemów przetwarzania danych. Wiele starszych systemów powstało przed erą chmury, a dla usług o specyficznych wymaganiach, których istniejące rozwiązania chmurowe nie są w stanie spełnić, systemy wewnętrzne pozostają niezbędne. Na przykład aplikacje bardzo wrażliwe na opóźnienia, używane przykładowo w handlu wysokich częstotliwości, wymagają pełnej kontroli nad sprzętem.

Architektura systemów natywnych dla chmury

Rozwój chmury nie tylko doprowadził do pojawienia się nowego modelu ekonomicznego (subskrypcja usługi zamiast zakupu sprzętu i licencji na oprogramowanie), ale też wywarł znaczący wpływ na sposób implementacji systemów danych na poziomie technicznym. Termin **natywne dla chmury** opisuje architekturę zaprojektowaną tak, aby w pełni wykorzystywać możliwości usług chmurowych.

Niemal każde oprogramowanie, które można hostować samodzielnie, może być również oferowane jako usługa w chmurze. I rzeczywiście wiele popularnych systemów danych jest obecnie dostępnych w formie usług zarządzanych. Jednak systemy projektowane od początku jako natywne chmurowo mają wiele przewag: wyższą wydajność na tym samym sprzęcie, szybsze przywracanie stanu po awariach, możliwość błyskawicznego skalowania zasobów obliczeniowych w zależności od obciążenia oraz obsługę większych zbiorów danych [24, 25, 26]. Tabela 1.2 przedstawia przykładowe systemy obu typów.

Tabela 1.2. Przykłady samodzielnie hostowanych i natywnych dla chmury systemów bazodanowych

Kategoria	Samodzielnie hostowane	Natywne dla chmury
Operacyjne (OLTP)	MySQL, PostgreSQL, MongoDB	AWS Aurora [24], Azure SQL DB Hyperscale [25], Google Cloud Spanner
Analityczne (OLAP)	Teradata, ClickHouse, Spark	Snowflake [26], Google BigQuery, Azure Synapse Analytics

Warstwy usług chmurowych

Wiele samodzielnie hostowanych systemów danych wymaga spełnienia tylko bardzo prostych warunków: działają w standardowych systemach operacyjnych, takich jak Linux czy Windows, przechowują dane w plikach w systemie plików i komunikują się za pomocą standardowych protokołów sieciowych, takich jak TCP/IP. Niektóre narzędzia wymagają specjalistycznego sprzętu, na przykład procesorów graficznych (do uczenia maszynowego) czy interfejsów sieciowych RDMA, ale zazwyczaj oprogramowanie hostowane samodzielnie korzysta z typowych zasobów obliczeniowych: procesora, pamięci RAM, systemu plików i sieci IP.

Kup książkę Poleć książkę

W chmurze tego typu oprogramowanie można uruchamiać w środowisku IaaS, korzystając z jednej lub wielu maszyn wirtualnych (*instancji*) z określoną pulą procesorów, pamięci, dysków i przepustowości sieci. W porównaniu z maszynami fizycznymi instancje chmurowe uruchamiają się szybciej i oferują bardziej zróżnicowane zasoby, ale poza tym są podobne do tradycyjnych komputerów: możesz uruchomić na nich dowolne oprogramowanie, lecz sam odpowiadasz za zarządzanie nim.

Natomiast w usługach natywnych dla chmury ważne jest nie tylko korzystanie z zasobów obliczeniowych zarządzanych przez system operacyjny, ale także budowanie usług wyższego poziomu przez łączenie niskopoziomowych usług chmurowych. Oto przykłady:

- Usługi **pamięci obiektowej**, takie jak Amazon S3, Azure Blob Storage czy Cloudflare R2, pozwalają przechowywać duże pliki. Oferują bardziej ograniczone API niż typowy system plików (umożliwiają tylko podstawowy odczyt i zapis plików), ale mają tę zaletę, że ukrywają fakt, iż używane są fizyczne maszyny. Usługa automatycznie rozdziela dane między wiele maszyn, więc nie musisz martwić się o brak miejsca na dysku którejkolwiek z nich. Nawet jeśli niektóre maszyny lub ich dyski ulegną awarii, żadne dane nie zostaną utracone.
- Wiele innych usług jest z kolei zbudowanych na bazie pamięci obiektowej i innych usług chmurowych. Na przykład Snowflake to chmurowa analityczna baza danych (hurtownia danych), która używa S3 do przechowywania danych [26], a jeszcze inne usługi są oparte na Snowflake'u.

Jak zawsze przy analizowaniu abstrakcji w informatyce nie ma jednej właściwej odpowiedzi na pytanie, jaki poziom należy wybrać. Ogólna zasada jest taka, że abstrakcje wyższego poziomu są zwykle bardziej dostosowane do konkretnych zastosowań. Jeśli Twoje potrzeby są zgodne ze scenariuszami, dla których zaprojektowano określony system wyższego poziomu, skorzystanie z niego prawdopodobnie zapewni Ci niezbędne mechanizmy przy znacznie mniejszym nakładzie pracy, niż gdybyś miał zbudować rozwiązanie samodzielnie na podstawie systemów niższego poziomu. Jeżeli jednak nie istnieje system wysokiego poziomu spełniający Twoje wymagania, jedyną opcją pozostaje zbudowanie go samemu z komponentów niższego poziomu.

Rozdzielenie pamięci masowej i mocy obliczeniowej

W tradycyjnej informatyce pamięć dyskowa jest uważana za trwałą (zakładamy, że dane zapisane na dysku nie zostaną utracone). Aby zabezpieczyć się przed awarią pojedynczego dysku twardego, często stosuje się macierze RAID (ang. *Redundant Array of Independent Disks*), które przechowują kopie danych na kilku dyskach podłączonych do tej samej maszyny. Model RAID może być realizowany sprzętowo lub programowo przez system operacyjny. Odbywa się to w sposób niezauważalny dla aplikacji korzystających z systemu plików.

W chmurze instancje obliczeniowe (maszyny wirtualne) również mogą mieć podłączone dyski lokalne, jednak systemy natywne dla chmury traktują je raczej jak tymczasową pamięć podręczną, a nie jako miejsce długoterminowego przechowywania danych. Wynika to z faktu, że dysk lokalny może stać się niedostępny w wyniku awarii powiązanej instancji albo zastąpienia określonej instancji większą lub mniejszą (na innej maszynie fizycznej) w celu dostosowania systemu do zmian obciążenia.

Kup książkę Poleć książkę

Jako alternatywę dla dysków lokalnych usługi chmurowe oferują wirtualną pamięć dyskową, którą można odłączyć od jednej instancji i podłączyć do innej (Amazon EBS, dyski zarządzane w Azure oraz dyski trwałe w w Google Cloud). Taki wirtualny dysk nie jest fizycznym urządzeniem, lecz usługą chmurową obsługiwaną przez oddzielny zestaw maszyn. Taka usługa emuluje pracę dysku (*urządzenia blokowego*, gdzie każdy blok ma zazwyczaj rozmiar 4 KiB). Technologia ta umożliwia uruchamianie tradycyjnego oprogramowania opartego na dyskach w chmurze, jednak emulacja urządzenia blokowego wprowadza koszty, których można uniknąć w systemach projektowanych od podstaw z myślą o chmurze [24]. Opisane rozwiązanie sprawia również, że aplikacja jest bardzo wrażliwa na zakłócenia w pracy sieci, ponieważ każda operacja wejścia-wyjścia na wirtualnym urządzeniu blokowym jest w rzeczywistości wywołaniem sieciowym [27].

Aby rozwiązać ten problem, w usługach natywnych dla chmury zazwyczaj unika się stosowania dysków wirtualnych i zamiast nich stosuje się dedykowane usługi przechowywania danych zoptymalizowane na potrzeby konkretnych obciążeń. Usługi pamięci obiektowej, takie jak S3, są przeznaczone do długoterminowego przechowywania stosunkowo dużych plików o rozmiarach od kilkuset kilobajtów do kilku gigabajtów. Pojedyncze wiersze lub wartości przechowywane w bazie danych są przeważnie znacznie mniejsze, dlatego bazy danych w chmurze zwykle zarządzają mniejszymi wartościami w osobnej usłudze, a większe bloki danych (zawierające wiele pojedynczych wartości) przechowują w pamięci obiektowej [25, 28]. Sposoby realizacji tego podejścia omawiamy w rozdziale 4.

W tradycyjnej architekturze systemów ten sam komputer odpowiada zarówno za przechowywanie danych (dysk), jak i za obliczenia (CPU i RAM), natomiast w systemach natywnych dla chmury te dwie funkcje zostały w pewnym stopniu *rozdzielone* [9, 26, 29, 30]. Na przykład S3 tylko przechowuje pliki, a jeśli chcesz przeanalizować dane, musisz uruchomić kod odpowiedzialny za analizy poza S3. Oznacza to konieczność przesyłania danych przez sieć, co opisujemy szczegółowo w podrozdziale „Systemy rozproszone a systemy jednowęzłowe" w tym rozdziale.

Ponadto systemy natywne dla chmury są często *wielodostępne*, co oznacza, że zamiast udostępniać dedykowaną maszynę dla każdego klienta, dane i obliczenia wielu użytkowników są obsługiwane na tym samym współdzielonym sprzęcie przez tę samą usługę [31]. Wielodostępność może prowadzić do lepszego wykorzystania sprzętu, większej skalowalności i prostszego zarządzania z perspektywy dostawcy chmury, ale wymaga starannego projektowania, aby aktywność jednego klienta nie wpływała na wydajność ani bezpieczeństwo systemu dla innych użytkowników [32].

Operacje w erze chmury

Tradycyjnie osoby zarządzające infrastrukturą danych po stronie serwera nazywano **administratorami baz danych** lub **administratorami systemów**. W ostatnich latach wiele organizacji próbuje integrować role związane z tworzeniem oprogramowania i operacjami w zespołach, które odpowiadają zarówno za usługi backendowe, jak i infrastrukturę związaną z danymi. Filozofia *DevOps* wyznacza kierunek tego trendu. **Inżynierowie SRE** (ang. *site reliability engineers*) to implementacja tej idei opracowana w Google'u [33].

Kup książkę Poleć książkę

Rolą zespołów operacyjnych jest dbanie o niezawodne dostarczanie usług użytkownikom (w tym konfigurowanie infrastruktury i wdrażanie aplikacji) oraz utrzymywanie stabilnego środowiska produkcyjnego (w tym monitorowanie i diagnozowanie problemów mogących niekorzystnie wpływać na niezawodność). W systemach hostowanych lokalnie praca operacyjna zwykle wymaga wykonywania licznych zadań na poziomie pojedynczych maszyn. Zespoły odpowiadają na przykład za planowanie dostępności zasobów (monitorowanie dostępnej przestrzeni dyskowej i dodawanie dysków, zanim skończy się na nich miejsce), uruchamianie nowych maszyn, przenoszenie usług między maszynami oraz instalowanie poprawek systemu operacyjnego.

Wiele usług chmurowych udostępnia API, które ukrywa poszczególne maszyny faktycznie realizujące daną usługę. Na przykład w usłudze pamięci masowej w chmurze dyski o stałym rozmiarze są zastępowane *rozliczeniem według zużycia*. Pozwala to przechowywać dane bez wcześniejszego planowania ilości potrzebnego miejsca, a opłata naliczana jest na podstawie faktycznie wykorzystanej przestrzeni. Ponadto wiele usług chmurowych pozostaje wysoko dostępnych nawet w przypadku awarii pojedynczych maszyn (zobacz podrozdział „Niezawodność i odporność na błędy” w rozdziale 2.).

Przesunięcie nacisku z pojedynczych maszyn na usługi skutkuje zmianą roli zespołów operacyjnych. Nadrzędny cel, czyli zapewnienie niezawodności usług, pozostaje ten sam, ale do jego realizacji stosuje się nowe procesy i narzędzia. Filozofia DevOps/SRE kładzie większy nacisk na:

- automatyzację (preferowane są powtarzalne procesy zamiast jednorazowych zadań wykonywanych ręcznie);
- preferowanie tymczasowych maszyn wirtualnych i usług zamiast długo działających serwerów;
- możliwość częstego aktualizowania aplikacji;
- wyciąganie wniosków z incydentów;
- zachowywanie w organizacji wiedzy o systemie, nawet w obliczu rotacji pracowników [34].

Wraz z rozwojem usług chmurowych nastąpił podział ról: zespoły operacyjne w firmach udostępniających infrastrukturę specjalizują się w dostarczaniu niezawodnych usług dużej liczbie klientów, dzięki czemu użytkownicy tych usług mogą poświęcać minimum czasu i wysiłku na aspekty związane z infrastrukturą [35].

Klienci usług chmurowych nadal potrzebują działów operacyjnych, ale koncentrują się one na innych aspektach, takich jak wybór najbardziej odpowiedniej usługi do danego zadania, integracja różnych usług oraz migracja z jednej usługi do innej. Mimo że rozliczenia według zużycia eliminują potrzebę tradycyjnego planowania dostępności zasobów, nadal ważne jest, aby wiedzieć, jakie zasoby są wykorzystywane i do jakich celów. Pomaga to uniknąć marnowania pieniędzy na niepotrzebne zasoby w chmurze. Planowanie dostępności zasobów służy więc do planowania budżetu, a optymalizacja wydajności służy do optymalizacji kosztów [36]. Ponadto w usługach chmurowych obowiązują *limity zasobów* (na przykład maksymalna liczba procesów, które można uruchomić jednocześnie). Należy je poznać i uwzględniać w planach, zanim zostaną przekroczone [37].

Kup książkę Poleć książkę

Wdrożenie usługi chmurowej może być łatwiejsze i szybsze niż utrzymywanie własnej infrastruktury. Jednak także wtedy trzeba ponieść koszty nauki korzystania z usługi i ewentualnego radzenia sobie z jej ograniczeniami. Wyzwaniem staje się również integrowanie usług, ponieważ coraz więcej dostawców oferuje coraz bogatszy zestaw usług chmurowych dostosowanych do różnych scenariuszy [38, 39]. Procesy ETL (zobacz punkt „Hurtownie danych" w tym rozdziale) to tylko jeden z aspektów integracji. Także operacyjne usługi chmurowe trzeba ze sobą zintegrować. Obecnie brakuje standardów, które ułatwiłyby taką integrację, więc często niezbędne są znaczne nakłady pracy ręcznej.

Inne aspekty operacyjne, których nie można w pełni przenieść do usług chmurowych, to między innymi gwarantowanie bezpieczeństwa aplikacji i używanych przez nią bibliotek, zarządzanie interakcjami między własnymi usługami, monitorowanie obciążenia usług oraz śledzenie przyczyn problemów, takich jak spadki wydajności czy awarie. Chociaż chmura zmienia rolę działu operacyjnego, jest on tak samo potrzebny jak dotychczas.

Systemy rozproszone a systemy jednowęzłowe

System składający się z kilku maszyn komunikujących się przez sieć nazywamy **systemem rozproszonym**. Każdy z procesów działających w systemie rozproszonym nazywamy *węzłem*. Istnieje wiele powodów, dla których warto rozważyć architekturę rozproszoną:

Systemy z natury rozproszone

Jeśli aplikacja wymaga interakcji dwóch lub więcej użytkowników, z których każdy korzysta z własnego urządzenia, system jest z natury rozproszony. Komunikacja między urządzeniami musi wtedy odbywać się przez sieć.

Żądania między usługami chmurowymi

Jeśli dane są przechowywane w jednej usłudze, ale przetwarzane w innej, muszą zostać przesłane przez sieć z jednej usługi do drugiej. Systemy natywne dla chmury oraz mikrousługi (zobacz punkt „Mikrousługi i architektura bezserwerowa" w tym rozdziale) są więc z definicji rozproszone.

Odporność na błędy i wysoka dostępność

Jeśli aplikacja musi kontynuować pracę nawet wtedy, gdy jedna maszyna (lub kilka maszyn, sieć bądź całe centrum danych) przestanie działać, można zastosować więcej maszyn, aby zapewnić nadmiarowość. Gdy jedna maszyna zawiedzie, inna może przejąć jej funkcje. Zobacz podrozdział „Niezawodność i odporność na błędy" w rozdziale 2. i poświęcony replikacji rozdział 6.

Skalowalność

Jeśli ilość danych albo wymagania obliczeniowe rosną powyżej poziomu, jaki może zostać obsłużony przez jedną maszynę, można rozdzielić to obciążenie między wiele maszyn. Zobacz podrozdział „Skalowalność" w rozdziale 2.

Kup książkę Poleć książkę

Opóźnienie

Jeśli użytkownicy znajdują się na całym świecie, możesz chcieć umieścić serwery w różnych lokalizacjach, tak aby móc obsłużyć każdego użytkownika za pomocą najbliższego mu geograficznie centrum danych. Dzięki temu użytkownicy nie muszą czekać na przesył pakietów sieciowych dookoła świata. Zobacz podrozdział „Opis wydajności" w rozdziale 2.

Elastyczność

Jeśli aplikacja jest czasami obciążona, a czasami bezczynna, wdrożenie jej w chmurze pozwala skalować zasoby w górę lub w dół w zależności od zapotrzebowania. Dzięki temu firma płaci tylko za zasoby, których aktywnie używa. Trudniej osiągnąć podobny efekt na pojedynczej maszynie, która musi być przygotowana na obsługę maksymalnego obciążenia nawet w momentach, gdy wykorzystywanych jest niewiele zasobów.

Wykorzystanie specjalistycznego sprzętu

Różne części systemu mogą korzystać z różnych typów sprzętu dopasowanych do generowanego obciążenia. Na przykład pamięć obiektowa może wymagać maszyn z licznymi dyskami, ale niewielką liczbą procesorów, w systemie do analizy danych potrzebne będą maszyny z szybkimi procesorami i dużą ilością pamięci, ale bez dysków, a system uczenia maszynowego może korzystać z maszyn z procesorami graficznymi (które są znacznie wydajniejsze niż CPU przy trenowaniu głębokich sieci neuronowych i wykonywaniu innych zadań z obszaru uczenia maszynowego).

Zgodność z przepisami

W niektórych krajach obowiązują przepisy dotyczące rezydencji danych, wymagające, aby informacje o ich mieszkańcach były przechowywane i przetwarzane na terenie danego państwa [40]. Zakres tych regulacji jest różny. W niektórych państwach takie przepisy dotyczą tylko danych medycznych lub finansowych, w innych zaś zakres uwzględnianych informacji jest szerszy. Dlatego w usłudze z użytkownikami z kilku takich krajów trzeba przechowywać dane na serwerach w różnych lokalizacjach.

Zrównoważony rozwój

Jeśli firma może swobodnie decydować, gdzie i kiedy uruchamiać zadania, pozwala to wykonywać je w czasie i miejscu, gdzie dostępna jest energia odnawialna, jak też unikać ich pracy, gdy sieć energetyczna jest przeciążona. Pomaga to ograniczyć ślad węglowy i umożliwia korzystanie z taniej energii, kiedy jest dostępna [41, 42].

Te czynniki dotyczą zarówno usług, które samodzielnie tworzysz (kodu aplikacji), jak i usług obejmujących gotowe oprogramowanie (takie jak bazy danych).

Problemy z systemami rozproszonymi

Systemy rozproszone mają również wady. W każdym żądaniu i wywołaniu API przesyłanym przez sieć trzeba uwzględnić możliwość wystąpienia awarii. Połączenie sieciowe może zostać przerwane, usługa może być przeciążona lub ulec awarii, a w konsekwencji każde żądanie może przekroczyć limit czasu bez otrzymania odpowiedzi. W takiej sytuacji nie wiadomo, czy usługa otrzymała żądanie, a ponowna próba jego wysłania może nie być bezpieczna. Problemy te omawiamy szczegółowo w rozdziale 9.

Kup książkę Poleć książkę

Chociaż sieci w centrach danych są szybkie, wywołanie innej usługi wciąż jest znacznie wolniejsze niż wywołanie funkcji w tym samym procesie [43]. W trakcie pracy z dużymi wolumenami danych zamiast przesyłać dane z magazynu do oddzielnej maszyny używanej do wykonywania obliczeń, szybsze może być przeniesienie tych obliczeń na maszynę zawierającą dane [44]. Ponadto większa liczba węzłów nie zawsze oznacza wyższą szybkość. W niektórych scenariuszach prosty jednowątkowy program na jednym komputerze może działać dużo lepiej niż klaster z ponad 100 rdzeniami procesorów [45].

Diagnozowanie problemów w systemie rozproszonym jest często trudne. Jak ustalić źródło kłopotów, jeśli system wolno odpowiada? Techniki diagnozowania problemów w systemach rozproszonych są rozwijane w ramach modelu zapewniania *obserwowalności* [46, 47], co wiąże się ze zbieraniem danych o działaniu systemu i umożliwianiem uruchamiania zapytań w sposób pozwalający analizować zarówno wysokopoziomowe wskaźniki, jak i pojedyncze zdarzenia. Narzędzia do *tracingu*, takie jak OpenTelemetry, Zipkin i Jaeger, pozwalają monitorować, który klient uruchomił wywołanie dotyczące określonej operacji na danym serwerze i ile czasu zajęła obsługa poszczególnych wywołań [48].

Bazy udostępniają różne mechanizmy zapewniania spójności danych. Opisujemy je w rozdziałach 6. i 8. Jednak gdy każda usługa korzysta z osobnej bazy, utrzymanie spójności danych między różnymi usługami staje się problemem, który trzeba rozwiązać na poziomie aplikacji. Transakcje rozproszone, omówione w rozdziale 8., to jedna z technik zapewniania spójności, ale rzadko stosuje się je w kontekście mikrousług, ponieważ stoją w sprzeczności z celem uniezależnienia usług od siebie. Ponadto wiele baz danych nie obsługuje takich transakcji [49].

Z wszystkich tych powodów jeśli można wykonać dane zadanie na pojedynczej maszynie, często rozwiązanie to okazuje się znacznie prostsze i tańsze w porównaniu z konfigurowaniem systemu rozproszonego [22, 45, 50]. Procesory, pamięć i dyski stały się większe, szybsze i bardziej niezawodne. W połączeniu z jednowęzłowymi bazami danych, takimi jak DuckDB, SQLite i KùzuDB, sprawia to, że wiele zadań można obecnie wykonywać w pojedynczych węzłach. Temat ten omawiamy szczegółowo w rozdziale 4.

Mikrousługi i architektura bezserwerowa

Najpopularniejszym sposobem podziału systemu między wiele maszyn jest rozdzielenie go na klienty i serwery. W tym modelu klienty wysyłają żądania do serwerów. W komunikacji najczęściej wykorzystuje się protokół HTTP, co omawiamy w punkcie „Przepływ danych z użyciem usług w REST i RPC" w rozdziale 5. Ten sam proces może być jednocześnie serwerem (obsługującym przychodzące żądania) i klientem (wysyłającym żądania do innych usług).

Ten sposób budowania aplikacji tradycyjnie nazywano **SOA** (ang. *service-oriented architecture*). Ostatnio koncepcja ta została udoskonalona i przekształcona w **architekturę mikrousług** [51, 52]. W tej architekturze każda usługa ma jedno ściśle określone zadanie (na przykład w S3 jest to przechowywanie plików). Każda usługa udostępnia API, które klienty mogą wywoływać przez sieć, a za zarządzanie poszczególnymi usługami odpowiadają odrębne zespoły. Złożoną aplikację można więc podzielić na wiele współdziałających ze sobą usług, z których każdą zarządza osobny zespół. W systemach natywnych dla chmury bardzo często stosuje się podział na usługi, ale model oparty na usługach stosuje się również w systemach instalowanych lokalnie.

Kup książkę Poleć książkę

Podział złożonego oprogramowania na wiele usług ma kilka zalet: każdą usługę można aktualizować niezależnie, co obniża koszty koordynacji między zespołami, każdej usłudze można przydzielić potrzebne zasoby sprzętowe, szczegóły implementacji są ukrywane za API, a właściciele usługi mogą swobodnie zmieniać implementację bez wpływu na klienty. Jeśli chodzi o przechowywanie danych, zwykle każda usługa ma własne bazy danych i nie współdzieli ich z innymi usługami. Współdzielenie baz sprawiłoby, że cała struktura bazy stałaby się częścią API usługi. W takim modelu zmiana struktury byłaby trudna. Współdzielone bazy danych mogą również powodować, że zapytania jednej usługi negatywnie wpływają na wydajność innych komponentów.

Należy jednak zauważyć, że używanie wielu usług samo w sobie generuje złożoność. Każda usługa wymaga infrastruktury do wdrażania nowych wersji, dostosowywania przydzielonych zasobów sprzętowych do obciążenia, rejestrowania informacji w dziennikach, monitorowania stanu i alarmowania dyżurnego inżyniera po wykryciu problemu. Platformy do **orkiestracji**, na przykład Kubernetes, stały się popularnym narzędziem do wdrażania usług, ponieważ zapewniają podstawę niezbędnej infrastruktury. Testowanie usługi podczas jej rozwoju może być skomplikowane, gdyż trzeba również uruchomić wszystkie inne usługi, od których ona zależy.

Modyfikowanie API mikrousług bywa trudne. Klienty wywołujące API oczekują, że będzie ono zawierać określone pola. Developerzy często chcą dodawać lub usuwać pola w API w miarę zmieniających się potrzeb biznesowych, ale może to powodować awarie po stronie klientów. Co gorsza, takie awarie często są odkrywane dopiero na późnych etapach cyklu rozwoju, gdy zaktualizowane API usługi jest wdrażane w środowisku przedprodukcyjnym lub produkcyjnym. Standardy opisu API, takie jak OpenAPI i gRPC, pomagają zarządzać relacjami między API klienta i serwera. Takie standardy omawiamy szczegółowo w rozdziale 5.

Mikrousługi są przede wszystkim technicznym rozwiązaniem problemu zarządzania zespołami — pozwalają różnym grupom pracować niezależnie od siebie bez konieczności koordynacji działań. Jest to przydatne w dużych firmach, ale w małych, gdzie nie ma wielu zespołów, używanie mikrousług prawdopodobnie będzie niepotrzebnym obciążeniem, dlatego lepiej jest zaimplementować aplikację w najprostszy możliwy sposób [51].

Architektura bezserwerowa, czyli **FaaS** (ang. *function-as-a-service*), to kolejny model wdrażania usług, w którym zarządzanie infrastrukturą jest zlecane dostawcy chmury [32]. Gdy korzystasz z maszyn wirtualnych, musisz bezpośrednio decydować, kiedy uruchamiać lub wyłączać instancje. Natomiast w modelu bezserwerowym to dostawca chmury automatycznie przydziela i zwalnia zasoby sprzętowe w zależności od potrzeb. Odbywa się to na podstawie żądań napływających do usługi [53]. Wdrożenie modelu bezserwerowego pozwala przenieść większą część obciążenia operacyjnego na dostawców chmury i umożliwia elastyczne rozliczanie według zużycia zasobów, a nie według instancji.

Aby móc zaoferować takie korzyści, wielu dostawców infrastruktury bezserwerowej wprowadza limity czasu wykonywania funkcji i ograniczenia dla środowisk uruchomieniowych. Ponadto czas pierwszego wywołania funkcji bywa długi. Termin „bezserwerowa" jest nieco mylący, ponieważ każde wykonanie funkcji w modelu bezserwerowym odbywa się na serwerze (przy czym dla kolejnych wywołań mogą być używane różne instancje). Ponadto producenci infrastruktury,

Kup książkę Poleć książkę

na przykład hurtowni BigQuery i różnych wersji Kafki, stosują termin „bezserwerowa", aby zasygnalizować, że ich usługi automatycznie się skalują i umożliwiają rozliczanie według zużycia, a nie według instancji.

Przetwarzanie w chmurze i superkomputery

Chmura obliczeniowa nie jest jedynym sposobem budowania wielkoskalowych systemów komputerowych. Innym rozwiązaniem są **systemy obliczeniowe o wysokiej wydajności** (ang. *high-performance computing* — HPC) wykonywane na **superkomputerach**. Choć te obszary częściowo się pokrywają, systemy HPC często wiążą się z innymi priorytetami i technikami niż chmura obliczeniowa czy systemy w korporacyjnych centrach danych. Oto niektóre z tych różnic:

- Superkomputery są zazwyczaj wykorzystywane do wymagających obliczeniowo zadań naukowych, takich jak prognozowanie pogody, modelowanie klimatu, dynamika molekularna (symulowanie ruchu atomów i cząsteczek), złożone problemy optymalizacyjne oraz rozwiązywanie równań różniczkowych cząstkowych. Z kolei chmura obliczeniowa jest przeważnie używana do uruchamiania usług internetowych, systemów biznesowych i podobnych rozwiązań, które muszą obsługiwać żądania użytkowników z wysoką dostępnością.
- Superkomputery zwykle wykonują duże zadania wsadowe, które co jakiś czas zapisują stan obliczeń na dysku w postaci punktów kontrolnych. Gdy węzeł ulegnie awarii, typowym rozwiązaniem jest zatrzymanie całego klastra, naprawa uszkodzonego węzła, a następnie wznowienie obliczeń od ostatniego punktu kontrolnego [54, 55]. W usługach chmurowych zatrzymywanie całego klastra jest najczęściej niepożądane, ponieważ usługi muszą nieprzerwanie obsługiwać użytkowników z minimalnymi zakłóceniami.
- Węzły superkomputerów komunikują się zazwyczaj z wykorzystaniem pamięci współdzielonej i technologii RDMA (ang. *remote direct memory access*), co gwarantuje wysoką przepustowość i niskie opóźnienia, ale wymaga wysokiego poziomu zaufania między użytkownikami systemu [56]. W chmurze obliczeniowej sieć i maszyny są często współdzielone przez organizacje, które mają do siebie zaufanie. Wymaga to silniejszych zabezpieczeń, takich jak izolacja zasobów (na przykład z użyciem maszyn wirtualnych), szyfrowanie i uwierzytelnianie.
- Sieci w dużych chmurowych centrach danych są zwykle oparte na protokole IP i Ethernecie oraz działają w topologii Closa [9]. Pomaga to osiągnąć wysoki wskaźnik *bisection bandwidth* (jest to powszechnie stosowana miara ogólnej wydajności sieci) [54, 57]. Superkomputery często wykorzystują wyspecjalizowane topologie sieciowe, na przykład wielowymiarowe kraty i torusy [58], które zapewniają wyższą wydajność dla obciążenia roboczego systemu HPC o znanych wzorcach komunikacji.
- W rozwiązaniach chmurowych węzły mogą znajdować się w wielu regionach. W superkomputerach przeważnie przyjmuje się, że wszystkie węzły znajdują się blisko siebie.

Wielkoskalowe systemy analityczne czasami mają pewne cechy wspólne z superkomputerami, dlatego warto znać odpowiednie techniki z tej dziedziny, jeśli pracujesz w obszarze analityki. Jednak w tej książce skupiamy się głównie na usługach, które muszą być stale dostępne, co omawiamy w podrozdziale „Niezawodność i odporność na błędy" w rozdziale 2.

Kup książkę Poleć książkę

Systemy danych, prawo i społeczeństwo

Z tego rozdziału można wywnioskować, że architektura systemów danych jest kształtowana nie tylko przez cele i wymagania techniczne, ale również przez potrzeby ludzkie w organizacjach korzystających z tych systemów. Inżynierowie systemów danych coraz częściej mają świadomość, że służenie potrzebom własnej firmy to za mało. Ponosimy również odpowiedzialność wobec całego społeczeństwa.

Szczególną uwagę należy zwrócić na systemy przechowujące dane o ludziach i ich zachowaniach. Od 2018 roku *rozporządzenie o ochronie danych osobowych* (RODO) daje mieszkańcom wielu krajów europejskich większą kontrolę i prawa w zakresie ich danych osobowych. Podobne przepisy o ochronie prywatności zostały przyjęte w różnych innych krajach i regionach na całym świecie. Jedną z takich regulacji jest na przykład California Consumer Privacy Act (CCPA). Przepisy dotyczące sztucznej inteligencji, takie jak *rozporządzenie Parlamentu Europejskiego i Rady w sprawie sztucznej inteligencji*, nakładają dodatkowe ograniczenia na sposób wykorzystywania danych osobowych.

Ponadto nawet w obszarach niepodlegających bezpośrednio regulacjom rośnie świadomość wpływu systemów komputerowych na ludzi i społeczeństwo. Media społecznościowe zmieniły sposób, w jaki ludzie odbierają wiadomości, co wpływa na ich poglądy polityczne i może tym samym oddziaływać na wyniki wyborów. Zautomatyzowane systemy coraz częściej podejmują decyzje mające poważne konsekwencje dla jednostek. Decydują, kto powinien otrzymać kredyt lub ubezpieczenie, kogo zaprosić na rozmowę kwalifikacyjną czy kto zostanie uznany za podejrzanego o popełnienie przestępstwa [59].

Każdy, kto pracuje nad takimi systemami, ponosi współodpowiedzialność za rozważenie ich etycznego wpływu i zapewnienie zgodności z obowiązującym prawem. Nie każdy musi zostać ekspertem od regulacji i etyki, ale świadomość zasad prawnych i etycznych jest równie ważna jak na przykład podstawowa wiedza o systemach rozproszonych.

Względy prawne wpływają na same fundamenty projektowania systemów danych [60]. Na przykład RODO przyznaje użytkownikom prawo do usunięcia ich danych na żądanie (nazywane *prawem do bycia zapomnianym*). Jednak, jak wyjaśniamy w tej książce, w wielu systemach danych wykorzystuje się niemodyfikowalne elementy, na przykład dzienniki umożliwiające wyłącznie dodawanie wpisów. Jak więc umożliwić usunięcie danych ze środka pliku, który z założenia ma być niemodyfikowalny? Jak usunąć informacje, które zostały wykorzystane do wygenerowania zbiorów danych pochodnych (zobacz punkt „Systemy zapisu a systemy danych pochodnych" w tym rozdziale), takich jak dane treningowe dla modeli uczenia maszynowego? Te pytania prowadzą do nowych wyzwań inżynieryjnych.

Obecnie nie istnieją jeszcze jednoznaczne wytyczne określające, które konkretne technologie lub architektury systemów należy uznać za „zgodne z RODO". Rozporządzenie celowo nie narzuca konkretnych technologii, ponieważ postęp techniczny może prowadzić do szybkich zmian. Zamiast tego w aktach prawnych opisane są ogólne zasady podlegające interpretacji. Oznacza to, że nie ma prostych odpowiedzi na pytanie, jak zapewnić zgodność z przepisami o ochronie prywatności. W tej książce przyjrzymy się jednak niektórym technologiom właśnie w tym kontekście.

Kup książkę Poleć książkę

Zwykle przechowujemy dane, ponieważ uważamy, że ich wartość przewyższa koszty przechowywania. Należy jednak pamiętać, że koszty przechowywania to nie tylko rachunek za Amazon S3 czy inną usługę. Kalkulacja kosztów i korzyści powinna uwzględniać również ryzyko poniesienia odpowiedzialności prawnej i szkód wizerunkowych w przypadku wycieku lub przejęcia danych przez napastników, a także ryzyko kosztów sądowych i kar, jeśli się okaże, że przechowywanie i przetwarzanie danych odbywało się niezgodnie z prawem [50].

Organy administracji publicznej oraz organy ścigania i wymiaru sprawiedliwości mogą również zażądać od firm przekazania danych. Gdy istnieje ryzyko, że dane mogą ujawnić zachowania uznawane za przestępstwo (takie jak praktyki homoseksualne w kilku krajach Bliskiego Wschodu i Afryki lub starania o aborcję w kilku stanach USA), przechowywanie takich informacji stwarza realne zagrożenie bezpieczeństwa dla użytkowników. Wizyta w klinice aborcyjnej może zostać łatwo ustalona na podstawie danych lokalizacyjnych, a nawet za pomocą rejestru adresów IP użytkownika na przestrzeni czasu (wskazują one przybliżoną lokalizację).

Po uwzględnieniu wszystkich zagrożeń można dojść do wniosku, że niektórych danych nie warto przechowywać, dlatego należy je usunąć. Ta zasada **minimalizacji danych** stoi w sprzeczności z filozofią big data, która zakłada spekulatywne gromadzenie dużych ilości informacji na wypadek, gdyby okazały się przydatne w przyszłości [61]. Ale jest to zgodne z RODO, które nakazuje, aby dane osobowe były zbierane tylko w określonym, konkretnym celu. Nie należy ich później wykorzystywać w innych kontekstach ani przechowywać dłużej niż jest to konieczne do celów, dla których zostały zebrane [62].

Firmy również zwróciły uwagę na kwestie prywatności i bezpieczeństwa. Organizacje obsługujące karty kredytowe wymagają od podmiotów przetwarzających płatności przestrzegania rygorystycznych standardów branży kart płatniczych (ang. *payment card industry* — PCI). Takie podmioty przechodzą częste kontrole przez niezależnych audytorów w celu weryfikacji, czy stale zachowują zgodność z przepisami. Dostawcy oprogramowania również podlegają zwiększonej kontroli. Wielu nabywców wymaga obecnie od dostawców zgodności ze standardami Service Organization Control (SOC) Type 2. Podobnie jak w przypadku zgodności ze standardami PCI, dostawcy przechodzą audyty zewnętrzne w celu weryfikacji przestrzegania regulacji.

Na ogólnym poziomie ważne jest zachowanie równowagi między wymaganiami biznesowymi a potrzebami osób, których dane zbieramy i przetwarzamy. Jest to bardzo szerokie zagadnienie. W rozdziale 14. szczegółowo omawiamy kwestie etyki i zgodności z przepisami, w tym problemy związane z uprzedzeniami i dyskryminacją.

Podsumowanie

Tematem przewodnim tego rozdziału było zrozumienie kompromisów, czyli uświadomienie sobie, że na wiele pytań nie ma jednej właściwej odpowiedzi. Często istnieje kilka różnych podejść, z których każde ma określone wady i zalety. Przyjrzeliśmy się najważniejszym decyzjom wpływającym na architekturę systemów danych i wprowadziliśmy terminologię, która będzie potrzebna w dalszej części książki.

Kup książkę Poleć książkę

Zaczęliśmy od wprowadzenia rozróżnienia między systemami operacyjnymi (przetwarzanie transakcji, OLTP) a analitycznymi (OLAP). Omówiliśmy występujące między nimi różnice. Te rodzaje systemów nie tylko służą do zarządzania różnymi typami danych z odmiennymi wzorcami dostępu, ale także są przeznaczone do obsługi różnych grup użytkowników. Opisaliśmy też hurtownie i jeziora danych, gdzie za pomocą procesów ETL zapisywane są strumienie danych z systemów operacyjnych. W rozdziale 4. wyjaśniamy, że systemy operacyjne i analityczne często mają zupełnie odmienne wewnętrzne układy danych, co wynika z różnych typów zapytań, które muszą obsługiwać.

Następnie porównaliśmy stosunkowo nowe rozwiązanie, jakim są usługi chmurowe, z tradycyjnym podejściem opartym na lokalnym hostowaniu oprogramowania, które wcześniej dominowało w architekturze systemów danych. To, który z tych modeli okaże się bardziej opłacalny, w dużej mierze zależy od konkretnej sytuacji. Nie da się jednak zaprzeczyć, że rozwiązania natywne dla chmury wprowadzają duże zmiany w sposobie projektowania systemów danych, na przykład w zakresie rozdzielenia pamięci masowej od mocy obliczeniowej.

Systemy chmurowe są z natury rozproszone, więc pokrótce przyjrzeliśmy się kompromisom związanym z systemami rozproszonymi w porównaniu z używaniem pojedynczej maszyny. Są sytuacje, w których nie da się uniknąć zastosowania rozwiązań rozproszonych, ale zaleca się, by nie spieszyć się z ich tworzeniem, jeśli system może poprawnie działać na jednej maszynie. W rozdziale 9. szczegółowo analizujemy wyzwania związane z systemami rozproszonymi.

Na koniec wyjaśniliśmy, że architektura systemów danych zależy nie tylko od potrzeb biznesowych organizacji, ale także od przepisów chroniących prawa osób, których dane są przetwarzane. Jest to aspekt, który wielu inżynierów często ignoruje. Sposób przekładania wymogów prawnych na implementacje techniczne nie został jeszcze dobrze zdefiniowany, ale ważne jest, by mieć tę kwestię na uwadze w trakcie lektury dalszej części tej książki.

Bibliografia

[1] R.T. Kouzes i in., *The Changing Paradigm of Data-Intensive Computing*, „IEEE Computer", t. 42, nr 1, styczeń 2009, doi:10.1109/MC.2009.26.

[2] M. Kleppmann i in., „Local-first software: you own your data, in spite of the cloud", [w:] *2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, październik 2019, doi:10.1145/3359591.3359737.

[3] J. Reis i M. Housley, *Fundamentals of Data Engineering,* O'Reilly Media, 2022, ISBN: 9781098108304.

[4] R.P. Machado i H. Russa, *Analytics Engineering with SQL and dbt*, O'Reilly Media, 2023, ISBN: 9781098142384.

[5] E.F. Codd, S.B. Codd i C.T. Salley, *Providing OLAP to User-Analysts: An IT Mandate*, E. F. Codd Associates, 1993, zarchiwizowano pod adresem *https://perma.cc/RKX8-2GEE*.

Kup książkę Poleć książkę

[6] Ch. Soman i N. Pawar, *Comparing Three Real-Time OLAP Databases: Apache Pinot, Apache Druid, and ClickHouse*, *startree.ai*, kwiecień 2023, zarchiwizowano pod adresem *https://perma.cc/8BZP-VWPA*.

[7] S. Chaudhuri i U. Dayal, *An Overview of Data Warehousing and OLAP Technology*, „ACM SIGMOD Record", t. 26, nr 1, s. 65 – 74, marzec 1997, doi:10.1145/248603.248616.

[8] F. Özcan, Y. Tian i P. Tözün, „Hybrid Transactional/Analytical Processing: A Survey", [w:] *ACM International Conference on Management of Data* (SIGMOD), maj 2017, doi:10.1145/3035918.3054784.

[9] A. Prout i in., *Cloud-Native Transactions and Analytics in SingleStore*, [w:] *International Conference on Management of Data* (SIGMOD), czerwiec 2022, doi:10.1145/3514221.3526055.

[10] Ch. Zhang i in., *HTAP Databases: A Survey*, „IEEE Transactions on Knowledge and Data Engineering", t. 36, nr 11, kwiecień 2024, doi:10.1109/TKDE.2024.3389693.

[11] M. Stonebraker i U. Çetintemel, „'One Size Fits All': An Idea Whose Time Has Come and Gone", [w:] *21st International Conference on Data Engineering* (ICDE), kwiecień 2005, doi:10.1109/ICDE.2005.1.

[12] J. Cohen i in., *MAD Skills: New Analysis Practices for Big Data*, „Proceedings of the VLDB Endowment", t. 2, nr 2, s. 1481 – 1492, sierpień 2009, doi:10.14778/1687553.1687576.

[13] D. Olteanu, *The Relational Data Borg is Learning*, „Proceedings of the VLDB Endowment", t. 13, nr 12, sierpień 2020, doi:10.14778/3415478.3415572.

[14] M. Bornstein, M. Casado i J. Li, *Emerging Architectures for Modern Data Infrastructure: 2020*, future.a16z.com, październik 2020, zarchiwizowano pod adresem *https://perma.cc/LF8W-KDCC*.

[15] R. Hai i in., *Data Lakes: A Survey of Functions and Systems*, „IEEE Transactions on Knowledge and Data Engineering" (TKDE), t. 35, nr 12, s. 12571 – 12590, grudzień 2023, doi:10.1109/TKDE.2023.3270101.

[16] M. Fowler, *DataLake*, martinfowler.com, luty 2015, zarchiwizowano pod adresem *https://perma.cc/4WKN-CZUK*.

[17] B. Johnson i J. Adler, *The Sushi Principle: Raw Data Is Better*, Strata+Hadoop World, luty 2015.

[18] DataKitchen, Inc., *The DataOps Manifesto*, dataopsmanifesto.org, 2017, zarchiwizowano pod adresem *https://perma.cc/3F5N-FUQ4*.

[19] T. Manohar, *What is Reverse ETL: A Definition & Why It's Taking Off*, *hightouch.io*, listopad 2021, zarchiwizowano pod adresem *https://perma.cc/A7TN-GLYJ*.

[20] C. Fournier, *Why is it so hard to decide to buy?*, skamille.medium.com, lipiec 2021, zarchiwizowano pod adresem *https://perma.cc/6VSG-HQ5X*.

Kup książkę Poleć książkę

[21] D. Heinemeier Hansson, *Why we're leaving the cloud*, world.hey.com, październik 2022, zarchiwizowano pod adresem *https://perma.cc/82E6-UJ65*.

[22] N. Badizadegan, *Use One Big Server*, specbranch.com, sierpień 2022, zarchiwizowano pod adresem *https://perma.cc/M8NB-95UK*.

[23] S. Yegge, *Dear Google Cloud: Your Deprecation Policy is Killing You*, steveyegge.medium.com, sierpień 2020, zarchiwizowano pod adresem *https://perma.cc/KQP9-SPGU*.

[24] A. Verbitski i in., „Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases", [w:] *ACM International Conference on Management of Data* (SIGMOD), s. 1041 – 1052, maj 2017, doi:10.1145/3035918.3056101.

[25] P. Antonopoulos i in., „Socrates: The New SQL Server in the Cloud", [w:] *ACM International Conference on Management of Data* (SIGMOD), s. 1743 – 1756, czerwiec 2019, doi:10.1145/3299869.3314047.

[26] M. Vuppalapati i in., *Building An Elastic Query Engine on Disaggregated Storage*, 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI), luty 2020.

[27] N. Van Wiggeren, *The Real Failure Rate of EBS*, planetscale.com, marzec 2025, zarchiwizowano pod adresem *https://perma.cc/43CR-SAH5*.

[28] C. Breck, *Predicting the Future of Distributed Systems*, blog.colinbreck.com, sierpień 2024, zarchiwizowano pod adresem *https://perma.cc/K5FC-4XX2*.

[29] G. Shapira, *Compute-Storage Separation Explained*, thenile.dev, styczeń 2023, zarchiwizowano pod adresem *https://perma.cc/QCV3-XJNZ*.

[30] R. Murthy i G. Goindi, *AlloyDB for PostgreSQL under the hood: Intelligent, database-aware storage*, *cloud.google.com*, maj 2022, zarchiwizowano pod adresem *https://web.archive.org/web/20220514021120/https://cloud.google.com/blog/products/databases/alloydb-for-postgresql-intelligent-scalable-storage*.

[31] J. Vanlightly, *The Architecture of Serverless Data Systems*, jack-vanlightly.com, listopad 2023, zarchiwizowano pod adresem *https://perma.cc/UDV4-TNJ5*.

[32] E. Jonas i in., *Cloud Programming Simplified: A Berkeley View on Serverless Computing*, arxiv.org, luty 2019.

[33] B. Beyer i in., *Site Reliability Engineering: How Google Runs Production Systems*, O'Reilly Media, 2016, ISBN: 9781491929124.

[34] T. Limoncelli, *The Time I Stole $10,000 from Bell Labs*, „ACM Queue", t. 18, nr 5, listopad 2020, doi:10.1145/3434571.3434773.

[35] Ch. Majors, *The Future of Ops Jobs*, *acloudguru.com*, sierpień 2020, zarchiwizowano pod adresem *https://perma.cc/GRU2-CZG3*.

Kup książkę Poleć książkę

[36] B. Cherkasky, (*Over)Pay As You Go for Your Datastore*, *medium.com*, wrzesień 2021, zarchiwizowano pod adresem *https://perma.cc/Q8TV-2AM2*.

[37] Sh. Kushchi, *Serverless Doesn't Mean DevOpsLess or NoOps*, thenewstack.io, luty 2023, zarchiwizowano pod adresem *https://perma.cc/3NJR-AYYU*.

[38] E. Bernhardsson, *Storm in the stratosphere: how the cloud will be reshuffled*, erikbern.com, listopad 2021, zarchiwizowano pod adresem *https://perma.cc/SYB2-99P3*.

[39] B. Stancil, *The data OS*, benn.substack.com, wrzesień 2021, zarchiwizowano pod adresem *https://perma.cc/WQ43-FHS6*.

[40] M. Korolov, *Data residency laws pushing companies toward residency as aservice*, csoonline.com, styczeń 2022, zarchiwizowano pod adresem *https://perma.cc/CHE4-XZZ2*.

[41] S. Borenstein, *Can Data Centers Flex Their Power Demand?*, energyathaas.wordpress.com, kwiecień 2025, zarchiwizowano pod adresem *https://perma.cc/MUD3-A6FF*.

[42] B. Acun i in., *Carbon Dependencies in Datacenter Design and Management*, „ACM SIGENERGY Energy Informatics Review", t. 3, nr 3, s. 21 – 26, doi:10.1145/3630614.3630619.

[43] K. Nath, *These are the numbers every computer engineer should know*, freecodecamp.org, wrzesień 2019, zarchiwizowano pod adresem *https://perma.cc/RW73-36RL*.

[44] J.M. Hellerstein i in., *Serverless Computing: One Step Forward, Two Steps Back*, Conference on Innovative Data Systems Research (CIDR), styczeń 2019.

[45] F. McSherry, M. Isard i D.G. Murray, *Scalability! But at What COST?*, 15th USENIX Workshop on Hot Topics in Operating Systems (HotOS), maj 2015.

[46] C. Sridharan, *Distributed Systems Observability: A Guide to Building Robust Systems*, raport, O'Reilly Media, maj 2018, zarchiwizowano pod adresem *https://perma.cc/M6JL-XKCM*.

[47] Ch. Majors, *Observability — A 3-Year Retrospective*, thenewstack.io, sierpień 2019, zarchiwizowano pod adresem *https://perma.cc/CG62-TJWL*.

[48] B.H. Sigelman i in., *Dapper, a Large-Scale Distributed Systems Tracing Infrastructure*, raport techniczny Google dapper-2010-1, kwiecień 2010, zarchiwizowano pod adresem *https://perma.cc/K7KU-2TMH*.

[49] R. Laigner i in., *Data management in microservices: State of the practice, challenges, and research directions*, „Proceedings of the VLDB Endowment", t. 14, nr 13, s. 3348 – 3361, wrzesień 2021, doi:10.14778/3484224.3484232.

[50] J. Tigani, *Big Data is Dead*, *motherduck.com*, luty 2023, zarchiwizowano pod adresem *https://perma.cc/HT4Q-K77U*.

[51] S. Newman, *Building Microservices*, wyd. II, O'Reilly Media, 2021, ISBN: 9781492034025.

[52] Ch. Richardson, *Microservices: Decomposing Applications for Deployability and Scalability*, infoq.com, maj 2014, zarchiwizowano pod adresem *https://perma.cc/CKN4-YEQ2*.

Kup książkę Poleć książkę

[53] M. Shahrad i in., *Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider*, USENIX Annual Technical Conference (ATC), lipiec 2020.

[54] L.A. Barroso, U. Hölzle i P. Ranganathan, *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, wyd. III, Morgan & Claypool Synthesis Lectures on Computer Architecture, październik 2018, doi:10.2200/S00874ED3V01Y201809CAC046.

[55] D. Fiala i in., „Detection and Correction of Silent Data Corruption for Large-Scale High-Performance Computing", [w:] *International Conference for High Performance Computing, Networking, Storage and Analysis* (SC), listopad 2012, doi:10.1109/SC.2012.49.

[56] A. Kornfeld Simpson i in., *Securing RDMA for High-Performance Datacenter Storage Systems*, 12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud), lipiec 2020.

[57] A. Singh i in., „Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network", [w:] *Annual Conference of the ACM Special Interest Group on Data Communication* (SIGCOMM), sierpień 2015, doi:10.1145/2785956.2787508.

[58] G.K. Lockwood, *Hadoop's Uncomfortable Fit in HPC*, glennklockwood.blogspot.co.uk, maj 2014, zarchiwizowano pod adresem *https://perma.cc/S8XX-Y67B*.

[59] C. O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown Publishing, 2016, ISBN: 9780553418811.

[60] S. Shastri, *Understanding and Benchmarking the Impact of GDPR on Database Systems*, „Proceedings of the VLDB Endowment", t. 13, nr 7, s. 1064 – 1077, marzec 2020, doi:10.14778/3384345.3384354.

[61] M. Fowler, *Datensparsamkeit*, martinfowler.com, grudzień 2013, zarchiwizowano pod adresem *https://perma.cc/R9QX-CME6*.

[62] Rozporządzenie Parlamentu Europejskiego i Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r. (Rozporządzenie o ochronie danych osobowych), „Official Journal of the European Union" L 119/1, maj 2016.

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Skorowidz

A

Kup książkę Poleć książkę

B

Kup książkę Poleć książkę

C

D

E

Kup książkę Poleć książkę

F

G

H

I

Kup książkę Poleć książkę

J

K

Kup książkę Poleć książkę

Kup książkę Poleć książkę

O

P

Kup książkę Poleć książkę

Kup książkę Poleć książkę

R

S

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Ś

T

U

Kup książkę Poleć książkę

V

W

X

Z

Kup książkę Poleć książkę

Ż

Kup książkę Poleć książkę

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Zrozum systemy danych — od fundamentów po architekturę na skalę globalną

W erze chmury obliczeniowej, mikroserwisów i eksplozji danych każda nowoczesna aplikacja musi się mierzyć z wyzwaniami skalowalności, niezawodności i wydajności. Drugie wydanie kultowej książki Martina Kleppmanna, uzpełnione o współautorstwo Chrisa Riccominiego, to kompleksowy przewodnik po architekturze systemów danych — od relacyjnych baz danych, przez rozproszone systemy NoSQL, po chmurowe hurtownie danych i przetwarzanie strumieniowe. Zaktualizowane wydanie uwzględnia najnowsze trendy: architekturę natywną dla chmury, indeksy wektorowe dla AI, silniki synchronizacji i oprogramowanie local-first.

Książka prowadzi czytelnika przez cały ekosystem systemów danych — od modeli przechowywania i indeksowania danych, przez replikację i sharding, po transakcje rozproszone i algorytmy uzgadniania konsensusu. Autorzy nie ograniczają się do opisu narzędzi — tłumaczą, dlaczego systemy działają tak, a nie inaczej, i jakie kompromisy kryją się za każdą decyzją architektoniczną. Osobne rozdziały zostały poświęcone przetwarzaniu wsadowemu i strumieniowemu, filozofii systemów sterowanych zdarzeniami i etycznym aspektom pracy z danymi osobowymi.

***Przetwarzanie danych w dużej skali* stało się biblią w dziedzinie systemów rozproszonych. Niemal każdy poważny specjalista, jakiego znam, posiada jej egzemplarz.**

Will Wilson, CEO, Antithesis Legacy Code

W książce:

- Modele danych i języki zapytań
- Silniki przechowywania danych
- Replikacja, sharding i transakcje
- Systemy rozproszone
- Przetwarzanie wsadowe i strumieniowe
- Architektura natywna dla chmury
- Etyka i prawo

Martin Kleppmann — profesor nadzwyczajny na Uniwersytecie Cambridge, wcześniej pracował jako inżynier w LinkedIn. Zajmuje się badaniem systemów rozproszonych i protokołów kryptograficznych.

Chris Riccomini — inżynier z piętnastoletnim doświadczeniem (PayPal, LinkedIn, WePay), współtwórca Apache Samza i SlateDB. Współautor książki *Brakujący plik README*.

Ta książka powinna być lekturą obowiązkową dla inżynierów oprogramowania.

Kevin Scott, CTO, Microsoft

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-3954-7
9 788328 939547
Cena: 149,00 zł