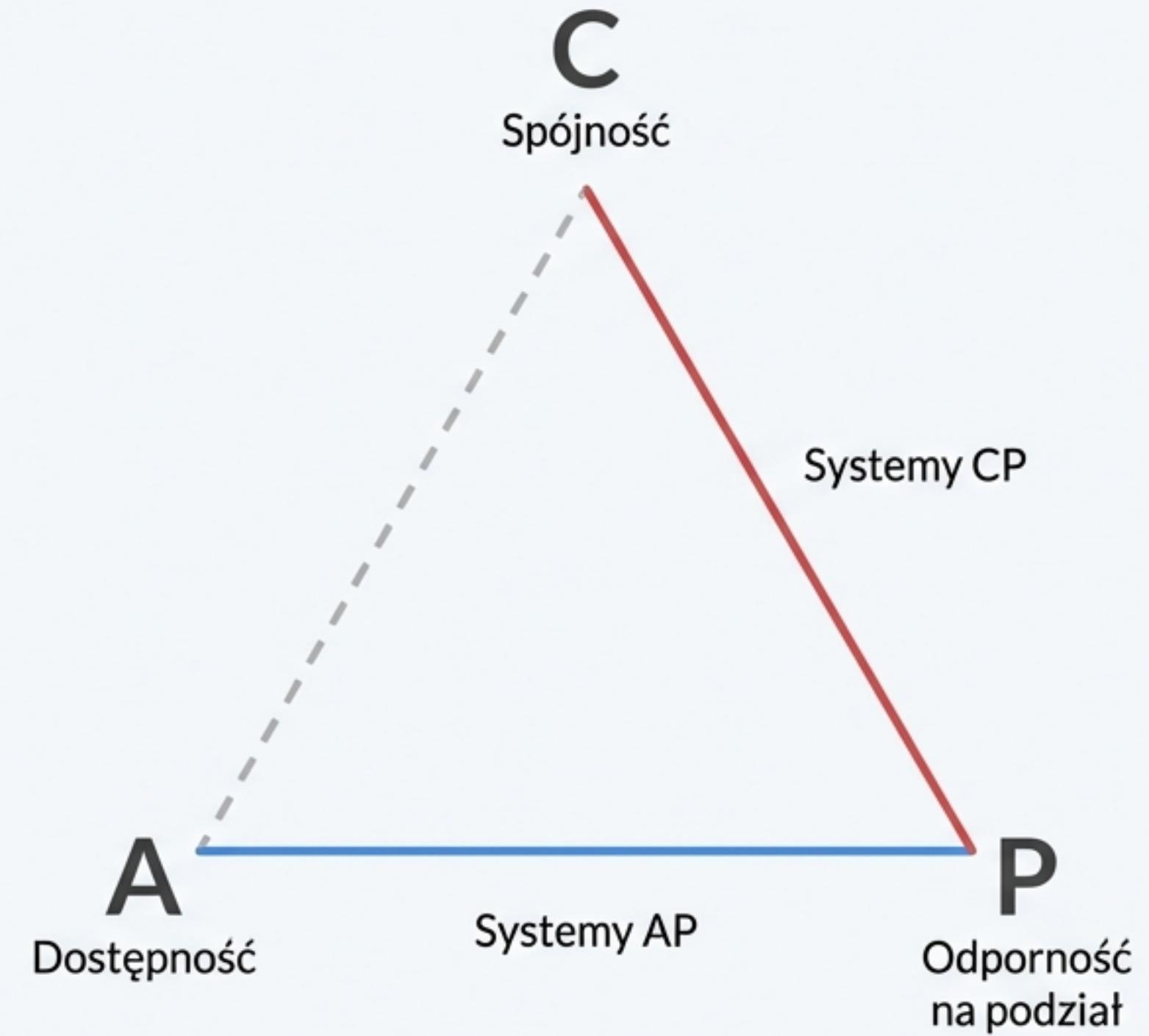


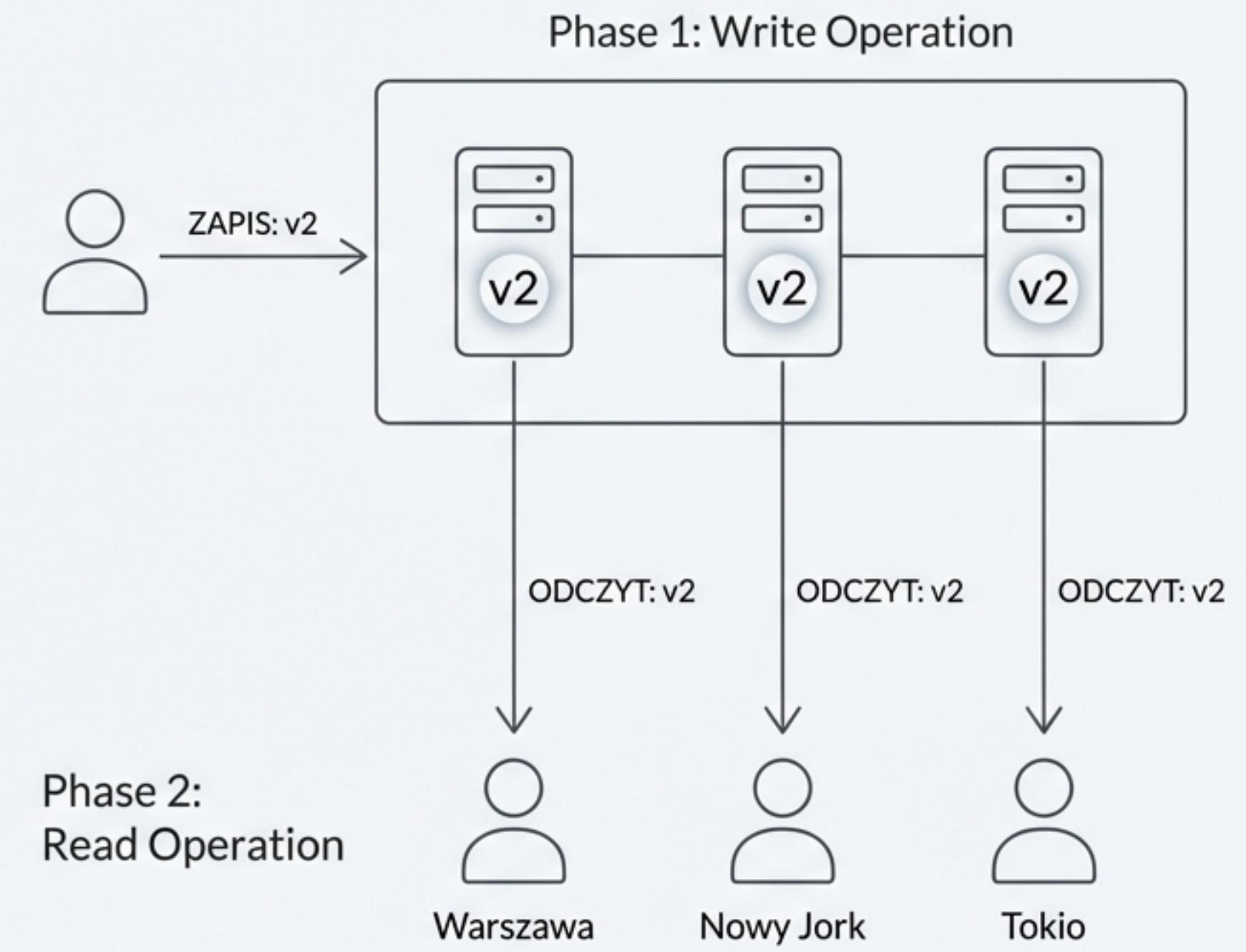
Twierdzenie CAP: Fundamentalny Kompromis

- Spójność (Consistency), Dostępność (Availability), Odporność na podział (Partition Tolerance) – w systemie rozproszonym można zagwarantować tylko dwie z tych trzech właściwości jednocześnie.
- W realnych systemach podziały sieci (partitions) są nieuniknione. To sprawia, że odporność na podział jest warunkiem koniecznym, a nie opcją do wyboru.
- W konsekwencji, podczas awarii sieci projektanci muszą dokonać trudnego wyboru: poświęcić Spójność (CP) na rzecz Dostępności, czy Dostępność (A) na rzecz Spójności.
- Sformułowane przez Erica Brewera, stało się kamieniem węgielnym w teorii i praktyce projektowania nowoczesnych systemów rozproszonych.



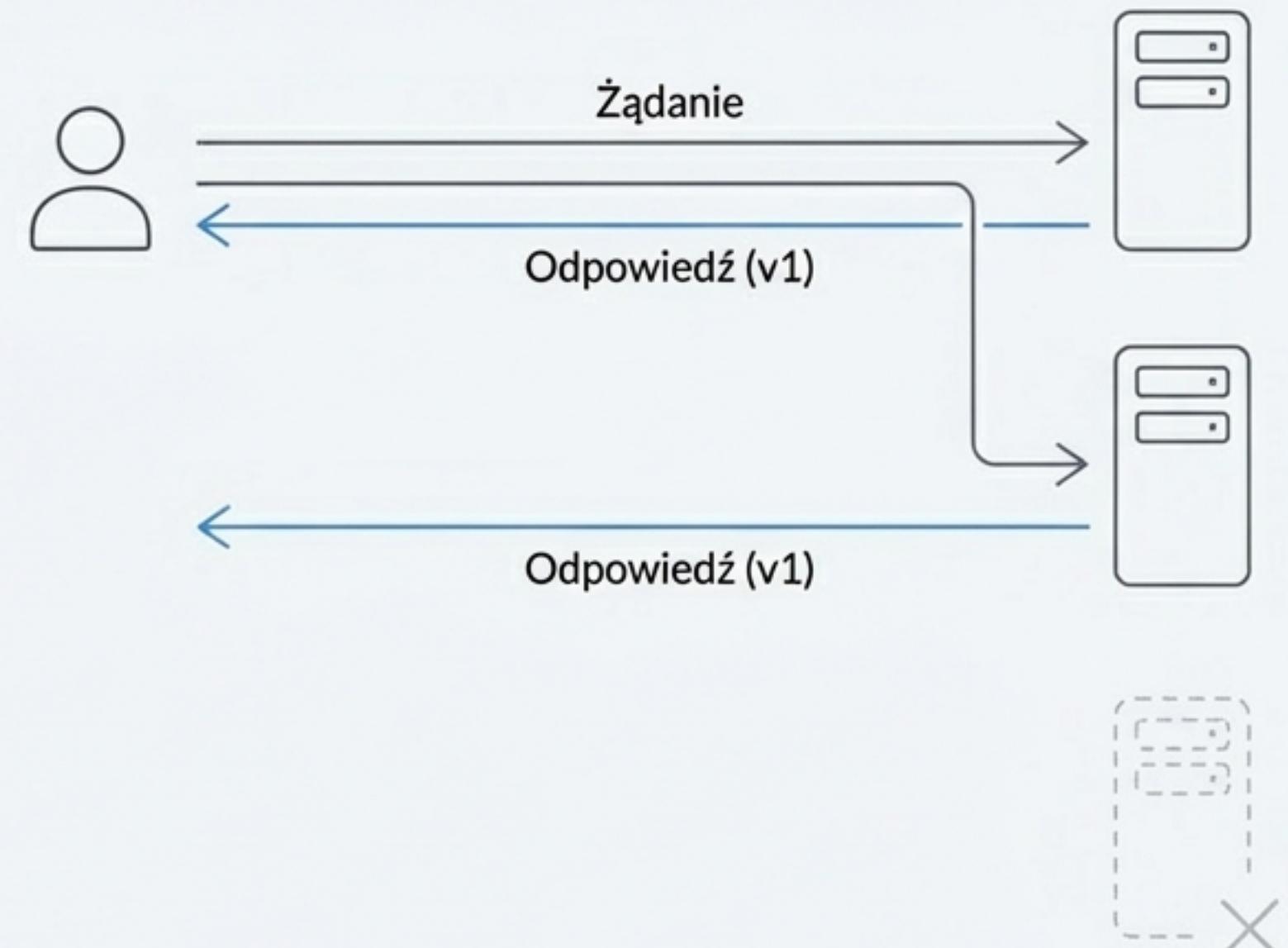
Spójność (C): Jedno Globalne Źródło Prawdy

- Gwarantuje, że każda operacja odczytu zwraca wynik ostatniej operacji zapisu. Wszyscy klienci widzą te same dane w tym samym czasie, niezależnie od tego, z którym węzłem się łączą.
- Z perspektywy użytkownika system zachowuje się jak monolityczna, pojedyncza maszyna.
- Wyobraźmy sobie użytkowników w Warszawie, Nowym Jorku i Tokio – wszyscy widzą dokładnie ten sam stan danych w tej samej chwili.
- *To jak jedna, magiczna tablica ogłoszeń. Gdy ktoś przypnie nową kartkę, wszyscy na świecie widzą ją natychmiast.*

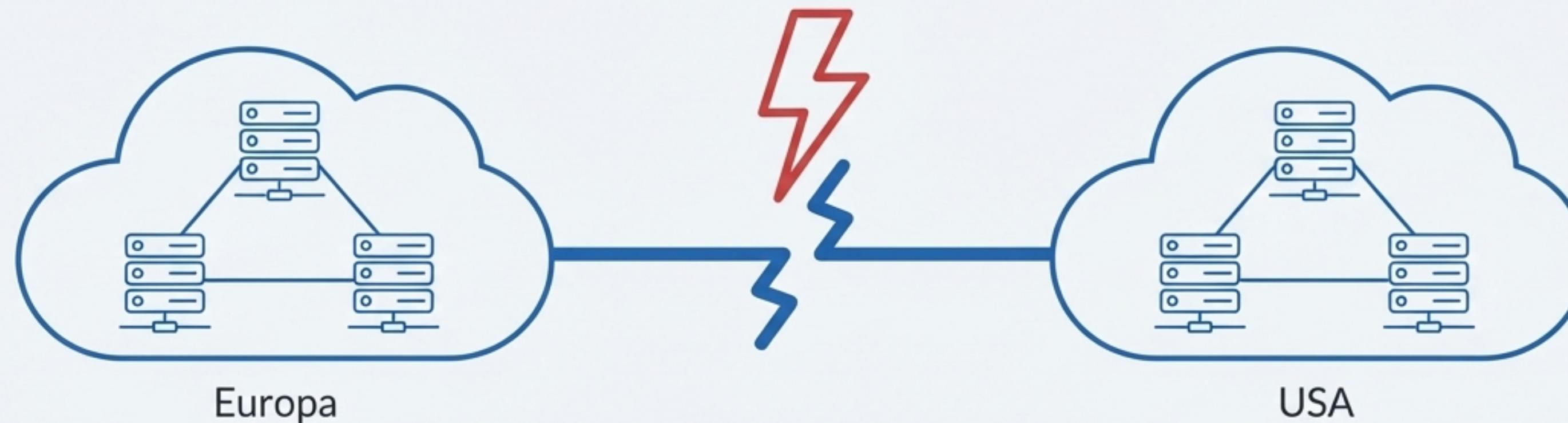


Dostępność (A): System Zawsze Odpowiada

- Każde żądanie skierowane do działającego węzła w systemie kończy się odpowiedzią. System nie może zwracać zwracać błędu informującego o niedostępności.
- Gwarantuje ciągłość działania usługi, nawet jeśli oznacza to, że zwrócone dane mogą być nieaktualne.
- Priorytetem jest to, by system zawsze był online dla użytkownika, a nie to, by dane były w 100% świeże.
- **METAFORA:** *To jak posiadanie lokalnych, podręcznych kopii ważnych dokumentów. Są zawsze pod ręką, ale mogą nie być najnowszą wersją z centralnego archiwum.*



Odporność na Podział (P): Przetrwanie Awarii Sieci



- System musi kontynuować działanie pomimo utraty (lub opóźnienia) dowolnej liczby wiadomości między węzłami.
- Podział sieci (network partition) to nie hipotetyczne zdarzenie, to codzienność w internecie: awarie routerów, przecięte kable podmorskie, błędy konfiguracji.
- W architekturze rozproszonej nie pytamy „czy” sieć zawiedzie, ale „kiedy” i „jak często”.
- Z tego powodu, Odporność na Podział nie jest opcją. Jest fundamentalnym wymogiem każdego realnego systemu rozproszonego.

Systemy CP: Spójność ponad Dostępnością

- W momencie wykrycia podziału sieci, system stawia na spójność danych. Aby uniknąć ryzyka niespójnych operacji, część systemu staje się niedostępna.
- System odrzuca żądania zapisu lub zwraca błędy, zamiast ryzykować podanie nieaktualnych lub sprzecznych danych.
- PRZYKŁAD: System rezerwacji biletów lotniczych. Aby zapobiec podwójnej sprzedaży tego samego samego miejsca, podczas awarii sieci system musi zablokować możliwość rezerwacji. Lepiej tymczasowo odmówić usługi niż sprzedać bilet, którego nie ma.

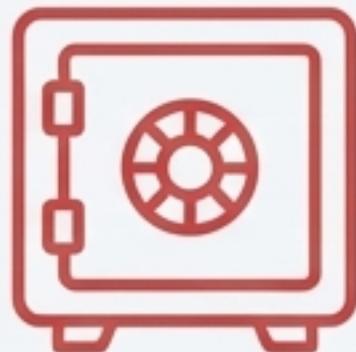


Systemy AP: Dostępność ponad Spójnością

- Podczas podziału sieci, system pozostaje dostępny dla operacji zapisu i odczytu. Każda partycja działa niezależnie.
- Akceptuje się ryzyko tymczasowej niespójności. Dane w odizolowanych częściach systemu mogą się różnić, prowadząc do konfliktów.
- System opiera się na modelu „Ostatecznej Spójności” (Eventual Consistency). Po ustaniu awarii, dane są synchronizowane, a konflikty automatycznie rozwiązywane.
- PRZYKŁAD: Koszyk w sklepie internetowym. System woli pozwolić na dodanie produktu do koszyka (nawet jeśli stan magazynowy jest nieaktualny) niż zablokować tysiącom użytkowników możliwość zakupów.



Dwa Światy: Bankowość kontra Media Społecznościowe



Systemy CP Spójność jest krytyczna

Przykłady: Systemy bankowe, transakcje giełdowe, kontrola ruchu lotniczego.

Logika: Nieprawidłowe saldo konta lub sprzedaż nieistniejących akcji jest niedopuszczalna. Tymczasowa niedostępność jest akceptowalnym kosztem utrzymania absolutnej poprawności.



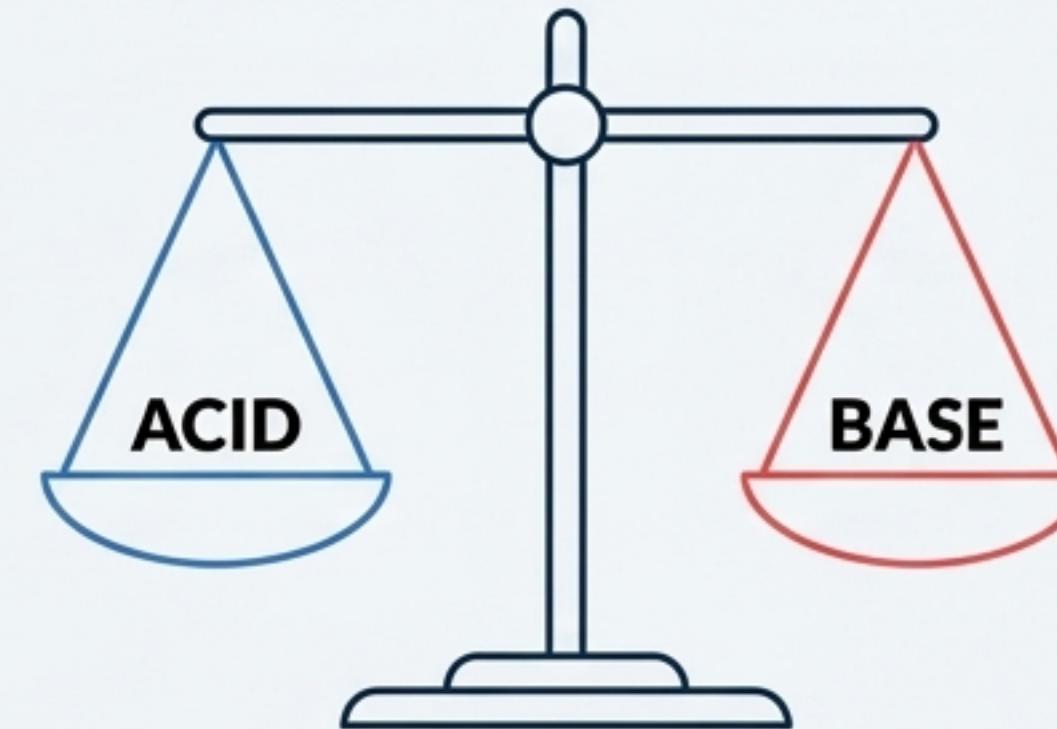
Systemy AP Dostępność jest kluczowa

Przykłady: Facebook, Instagram, YouTube, koszyki w e-commerce.

Logika: Użytkownik, który przez 15 minut nie może dodać zdjęcia, jest bardziej sfrustrowany niż ten, który zobaczy nowego „lajka” z sekundowym opóźnieniem.

PODSUMOWANIE: Ten sam serwis (np. Amazon) może używać modelu AP do przeglądania produktów i dodawania do koszyka, ale przełączać się na model CP na etapie finalizacji płatności.

ACID kontra BASE: Dwie Filozofie Projektowania Danych



ACID

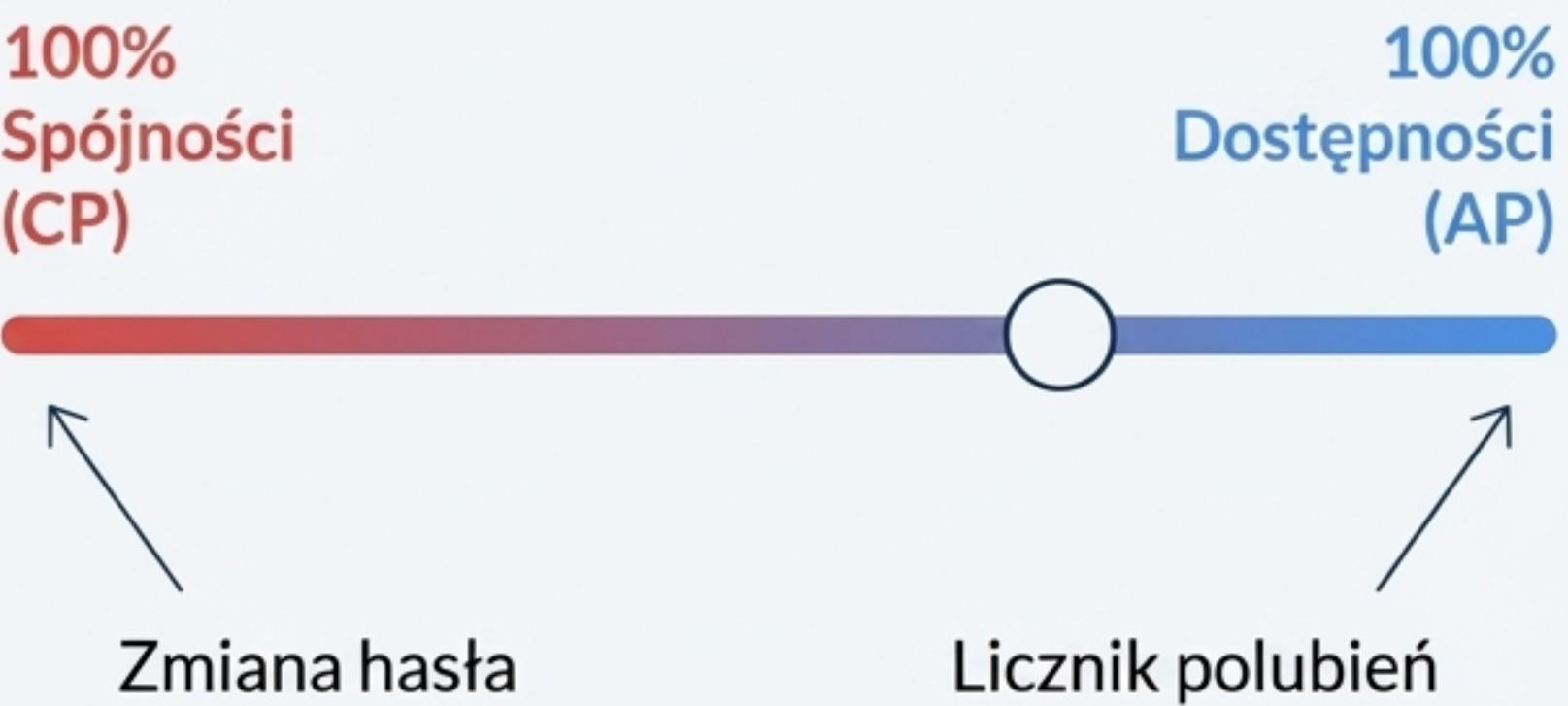
- **Reprezentuje:** Tradycyjne, relacyjne bazy danych (np. PostgreSQL, Oracle).
- **Gwarancje:** Silne, transakcyjne gwarancje spójności (Atomicity, Consistency, Isolation, Durability).
- **Model CAP:** Naturalnie skłania się ku modelowi CP.

BASE

- **Reprezentuje:** Nowoczesne, rozproszone bazy danych NoSQL (np. Cassandra, DynamoDB).
- **Gwarancje:** Kładzie nacisk na dostępność kosztem natychmiastowej spójności.
- **Akronim:**
 - **Basically Available:** System jest w zasadzie zawsze dostępny.
 - **Soft state:** Stan systemu może się zmieniać w czasie, nawet bez zewnętrznej interakcji.
 - **Eventually consistent:** System ostatecznie osiągnie spójność.
- **Model CAP:** Filozofia stojąca za systemami AP.

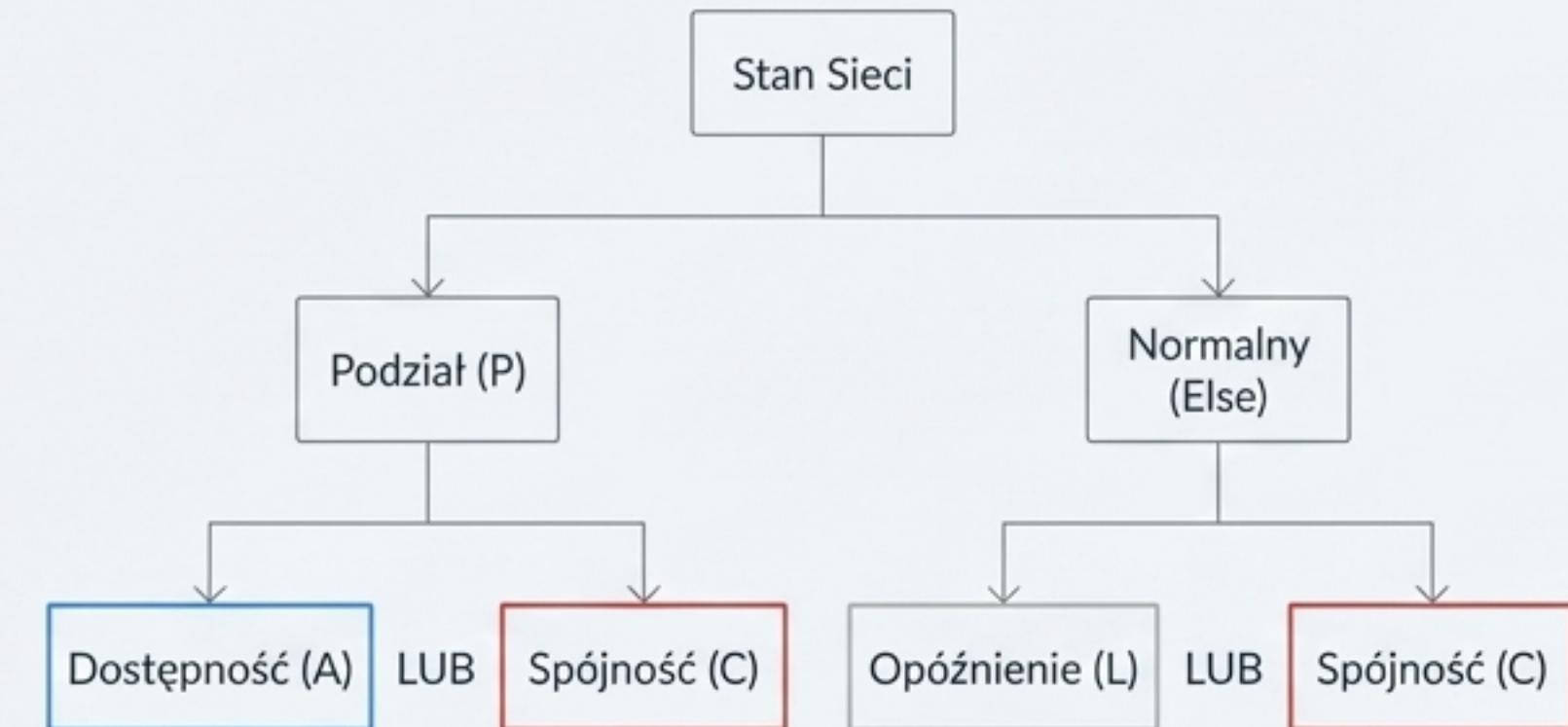
Poza Binarnym Wyborem: Spektrum Kompromisów

- W praktyce, systemy rzadko są „czysto” CP lub „czysto” AP. Projektanci poruszają się po spektrum, dobierając odpowiedni poziom gwarancji do konkretnego przypadku użycia.
- Różne operacje w ramach tego samego systemu mogą mieć różne charakterystyki.
- Odczyt liczby polubień pod zdjęciem może być ostatecznie spójny (AP), ale zmiana hasła użytkownika musi być silnie spójna (CP).
- Kompromis między wydajnością a spójnością istnieje nawet wtedy, gdy sieć działa poprawnie – nie jest to problem wyłącznie czasów awarii.



PACELC: Ewolucja Twierdzenia CAP

- PACELC rozszerza CAP, formalizując kompromis, który istnieje również w trakcie normalnego działania systemu (gdy nie ma podziału sieci).
- Twierdzenie można opisać jako serię wyborów „jeśli... to... w przeciwnym razie...”:
 - **If** there is a **Partition** (Jeśli jest podział), system musi wybrać między **Availability** (Dostępnością) a **Consistency** (Spójnością). (To jest oryginalne twierdzenie CAP).
 - **Else** (w przeciwnym razie), system musi wybrać między **Latency** (Opóźnieniem) a **Consistency** (Spójnością).
- Oznacza to wybór: czy chcemy szybszej odpowiedzi z potencjalnie nieaktualnymi danymi (niska latencja), czy wolniejszej odpowiedzi z gwarancją odczytu najnowszych danych (silna spójność)?



Kompromis jako Fundament Projektowania

Zrozumienie twierdzenia CAP i jego ewolucji to nie tylko akademickie ćwiczenie. To świadomość, że w cyfrowym świecie, którego niezawodność bierzemy za pewnik, każda usługa – od banku po ulubioną aplikację – jest zbudowana na fundamentalnym kompromisie między absolutną poprawnością a nieprzerwanym działaniem. Wybór ten definiuje charakter i granice technologii, z których korzystamy każdego dnia.

