GemStone GemFire
Rozproszony system zarządzania danymi

Systemy Rozproszone
12.11.2009

Karol Ruszczyk

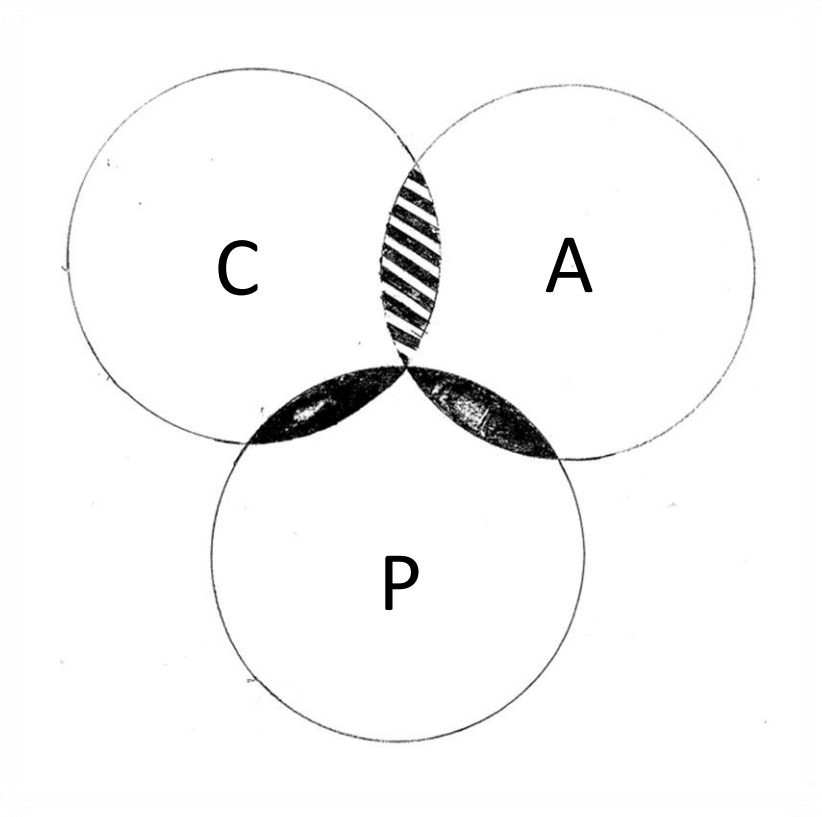
Agenda

- Wprowadzenie
- Zarys architektury GemFire
- Podstawowe topologie
- Strategie przechowywania danych

Wprowadzenie

Twierdzenie CAP

- Spójność
- Dostępność
- Odporność na podział



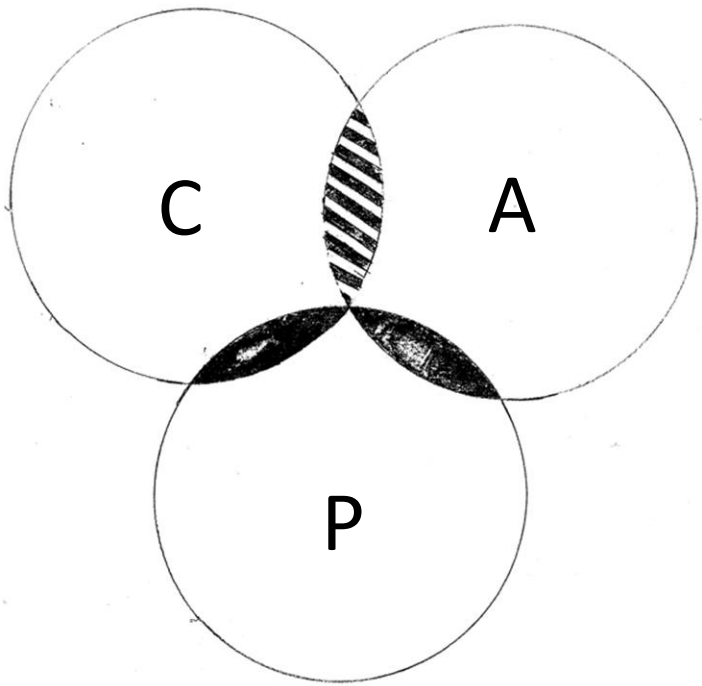
Twierdzenie CAP - konsekwencje

W przeciwieństwie do aplikacji webowych:

Spójność danych jest najważniejsza!

Dostępność – bardzo ważna!

… ale nie możemy zapominać o odporności na podział..



Myśl przewodnia

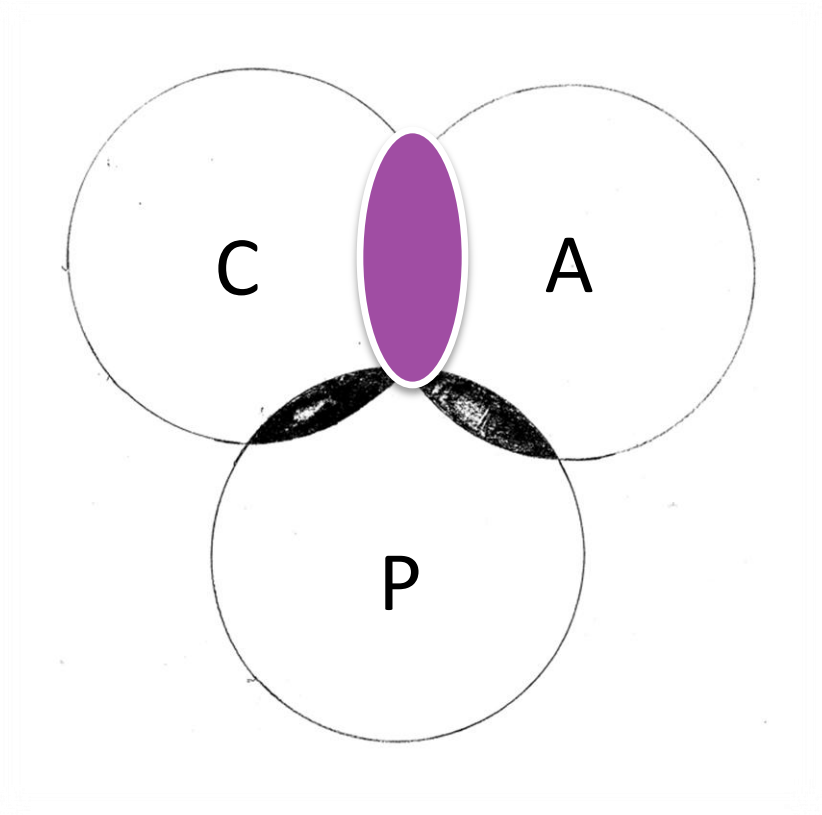
„Jak faworyzować spójność i dostępność a jednocześnie radzić sobie z sytuacjami awarii sieci?”

Dlaczego rozproszony cache? (a nie zwykła baza danych)

- Dane przechowywane w pamięci RAM
- Algorytmy i struktury danych budowane z myślą o tym, że dane są trzymane w pamięci a nie na dysku
- Mniejszy nacisk na długotrwałe przechowywanie danych

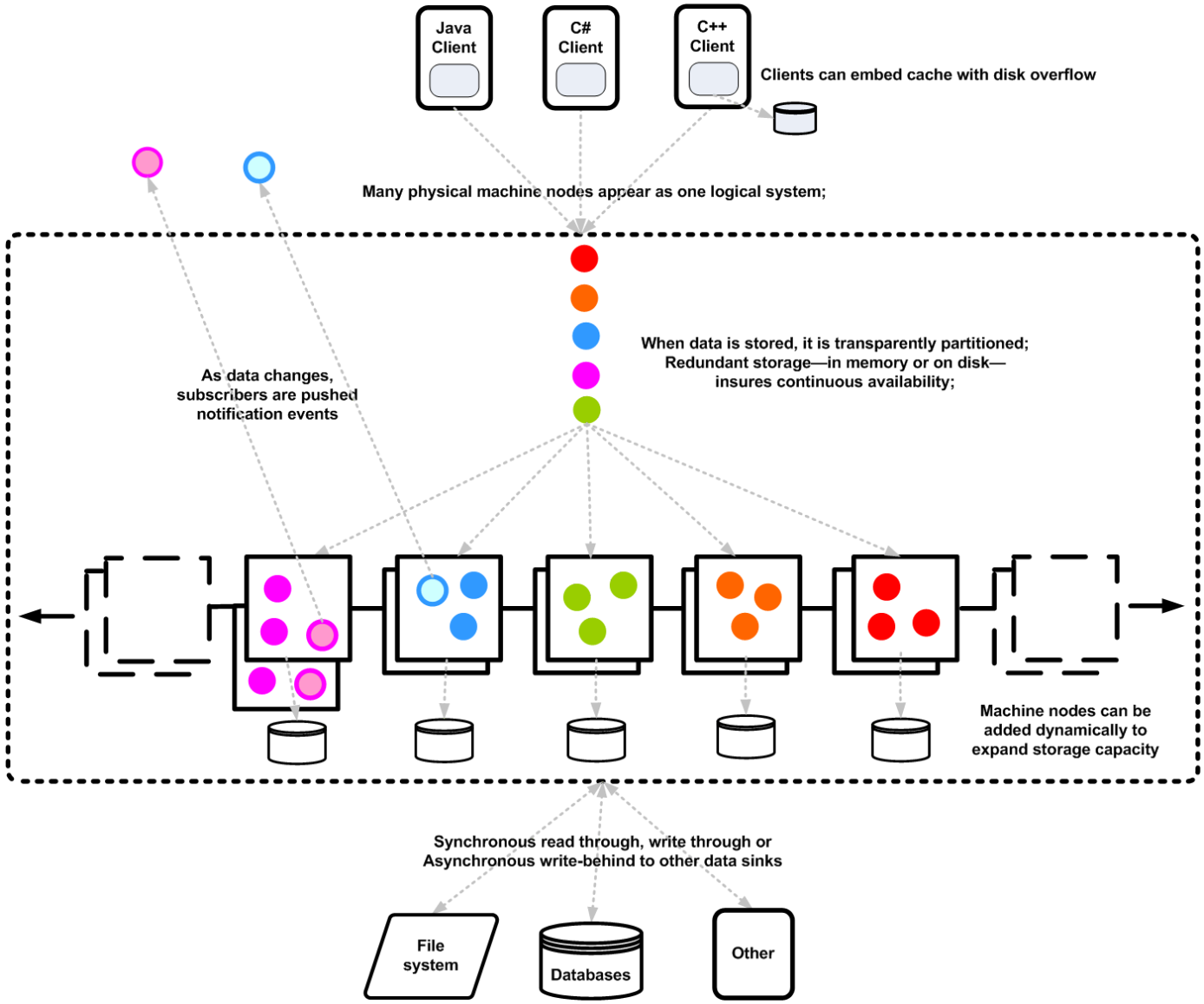
Najważniejsze aplikacje

- **GemStone GemFire**
- Oracle Coherence
- IBM ObjectGrid



Architektura

GemFire at a Glance



Słownik

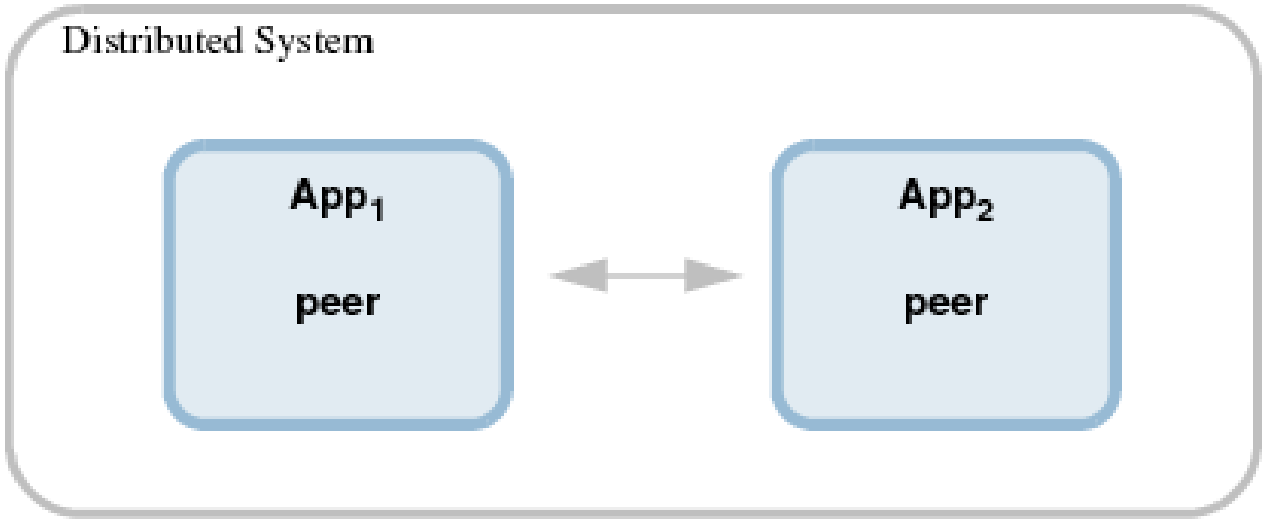
- Rozproszony system (**distributed system**)

  Członkowie, którzy tworzą logiczną grupę, mają uruchomioną instancję GemFire i są skonfigurowani aby komunikować się ze sobą
- Region (**region**)

  Logiczne zgrupowanie danych znajdujących się w systemie zarządzania danymi

Podstawowe topologie

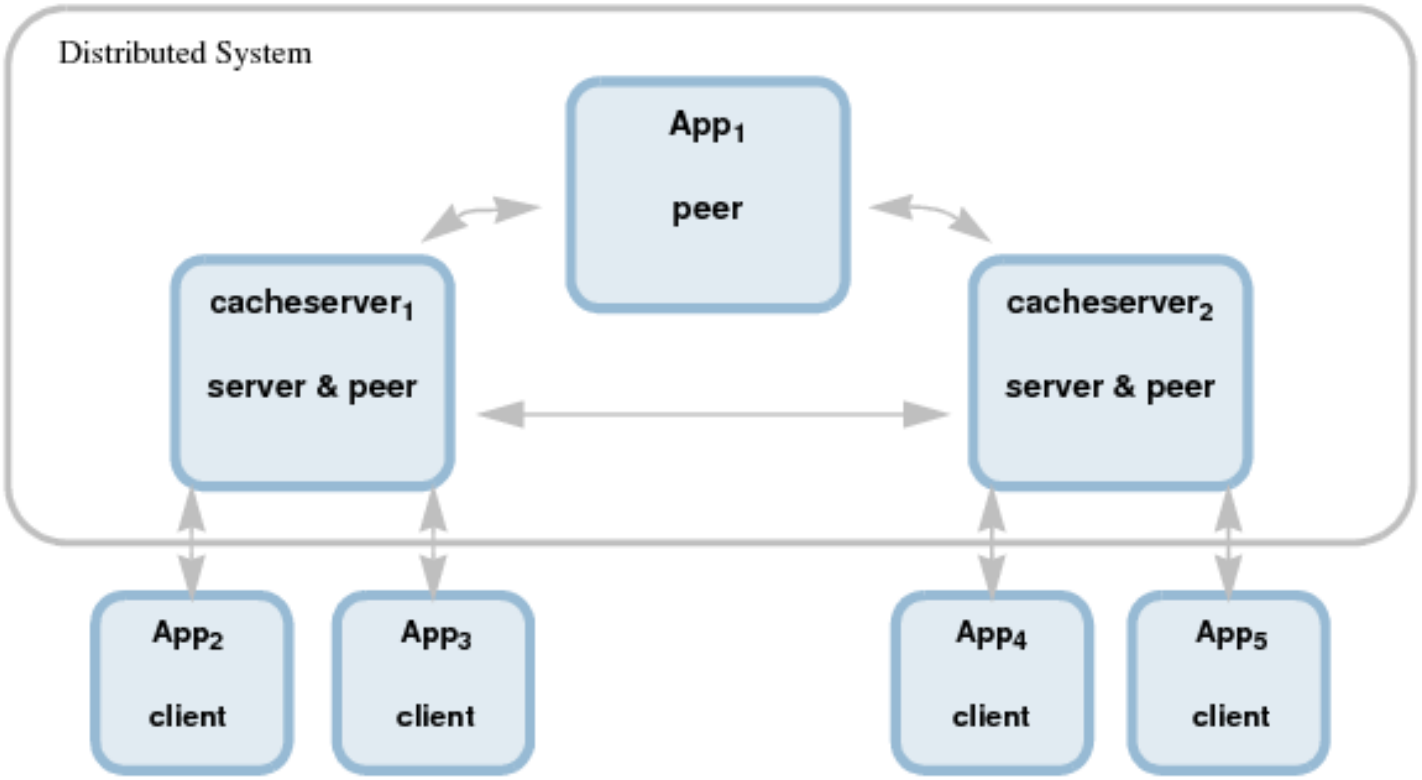
Peer-2-Peer



Peer-2-Peer

- Każdy członek jest równoprawny
- Komunikacja między członkami jest bezpośrednia
- *Region* istnieje dopóki zbiór członków jest niepusty
- Informacje o przyłączeniu/odłączeniu członka otrzymują są rozsyłane do wszystkich

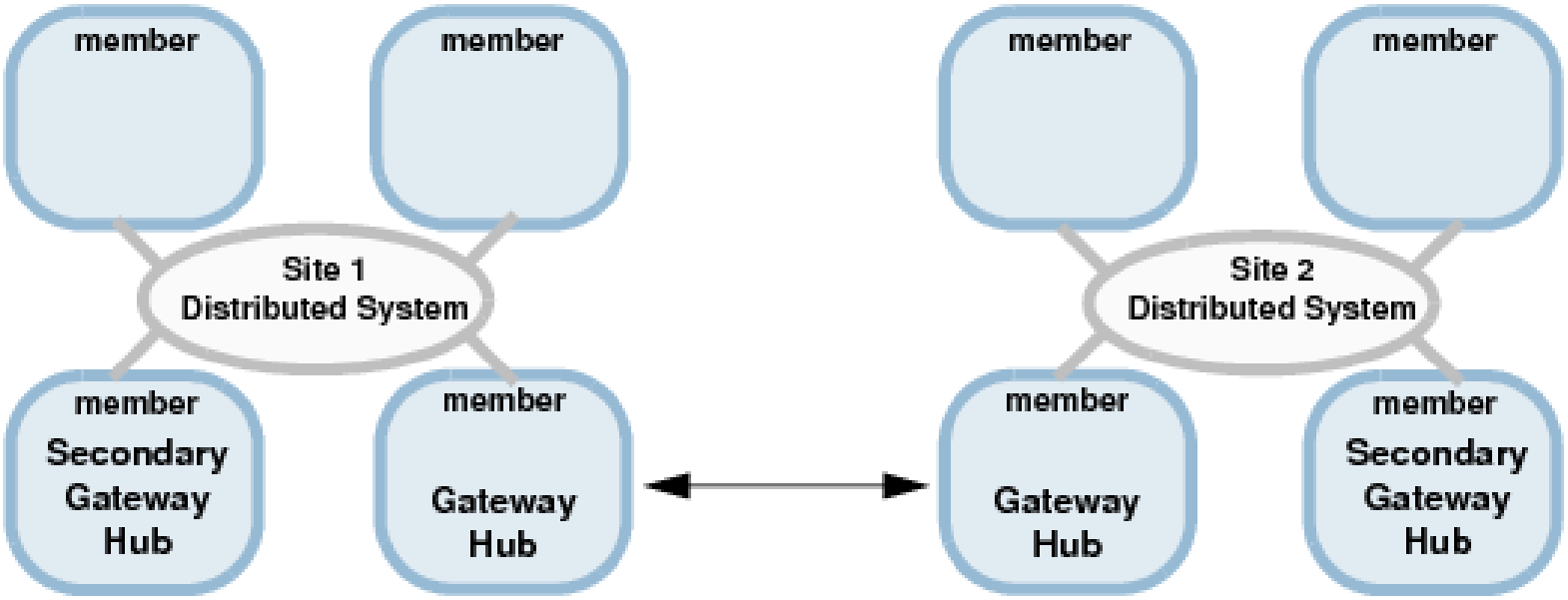
Klient - Serwer



Klient – Serwer

- Dynamiczne znajdowanie serwera przez klienta
- Łączenie serwerów w logiczne grupy
  - Równoważenie obciążenia
  - Automatyczne przełącznie w przypadku rozłaczenia serwera
- Automatyczne powiadamianie klienta o zmianach

Multi-site


member
member
Site 1
Distributed System
member
Secondary
Gateway
Hub
member
Gateway
Hub
member
member
Site 2
Distributed System
member
Gateway
Hub
member
Secondary
Gateway
Hub

Multi-site

- Połączenie dwóch lub więcej rozproszonych systemów
- W każdym systemie jest wyznaczony co najmniej jeden członek który służy jako *hub*
- Każdy z systemów jest niezależny i może działać gdy drugi jest niedostępny
- *Eventual consistency*

Strategie przechowywania danych

- empty
  - dane nigdy nie są przechowywane w lokalnej pamięci
- replicate
  - dane całego regionu przechowywane są w lokalnej pamięci
- persistent-replicate
  - jak replicate, dodatkowo umożliwia zrzut danych na dysk
- partition

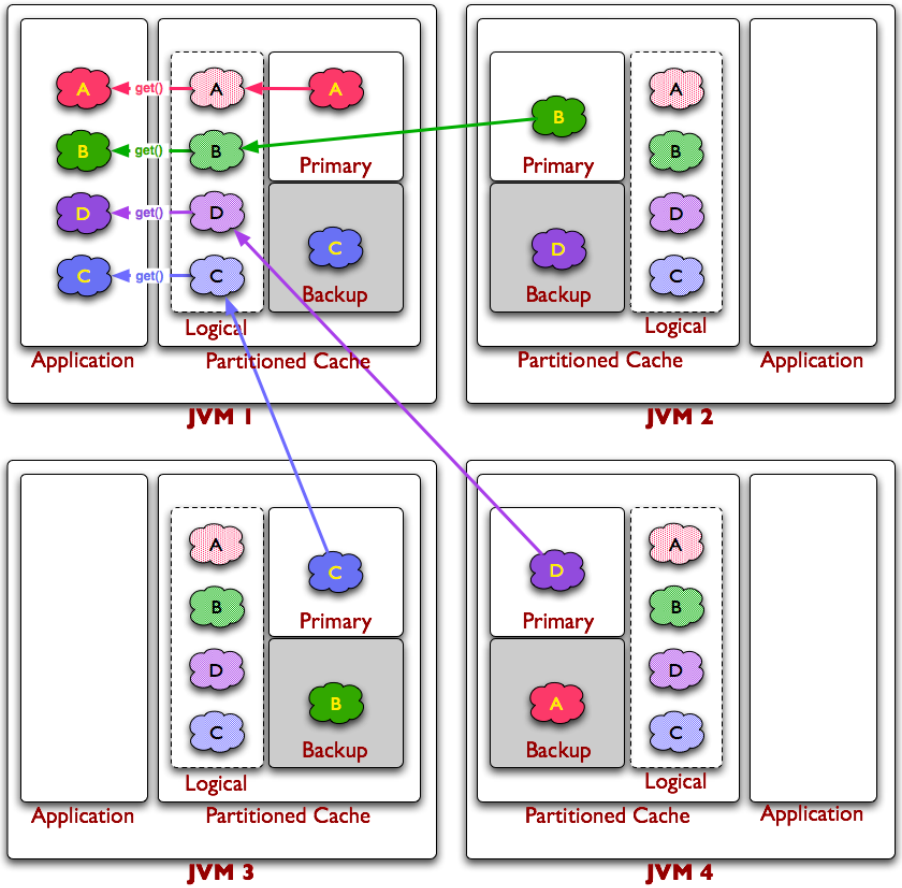
Regiony dzielone

(Partitioned Regions)

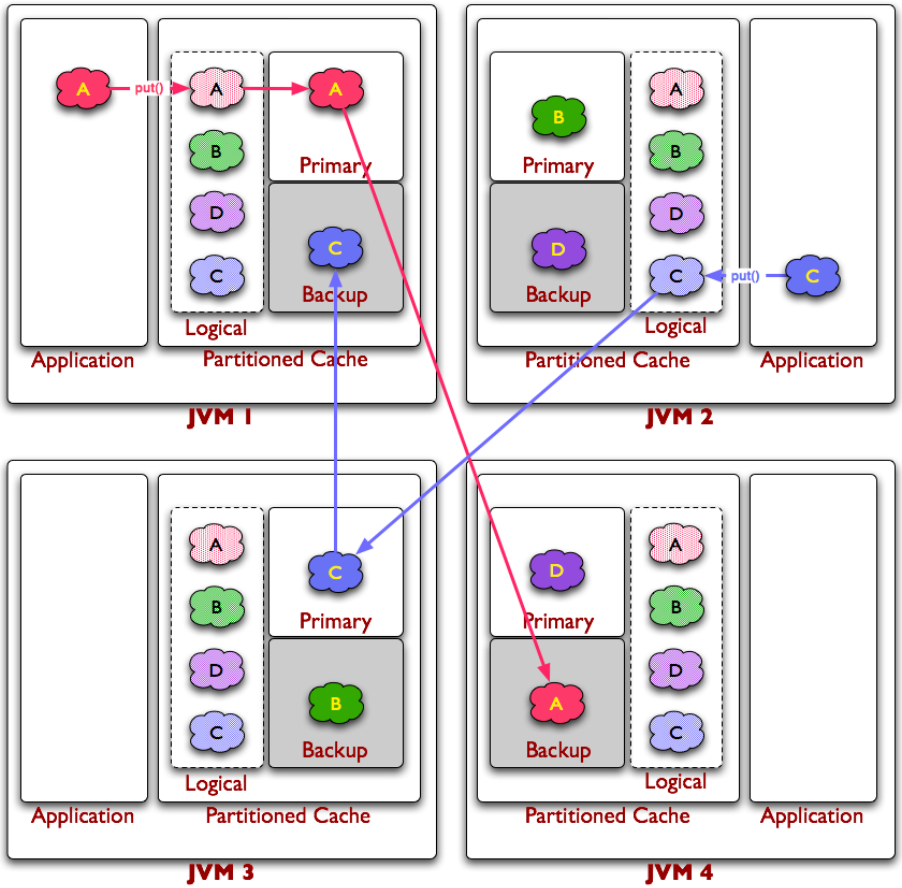
Co zrobić gdy..

- Mamy więcej danych niż jest w stanie pomieścić RAM pojedynczej maszyny
- Nie chcemy nic trzymać na dysku twardym

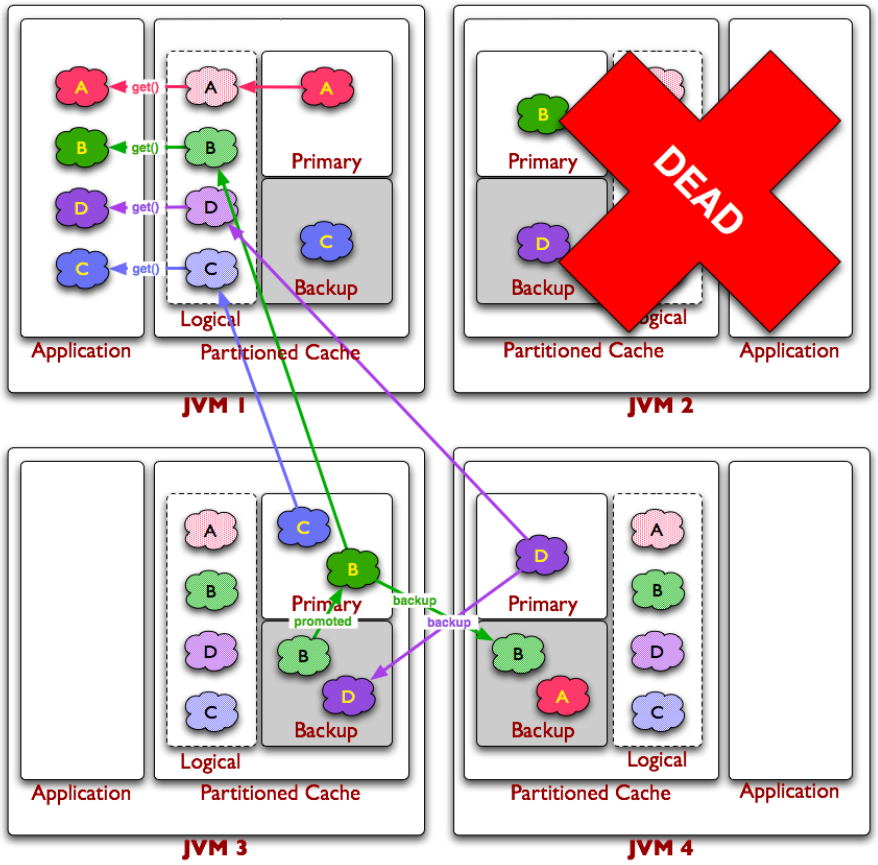
Region dzielony



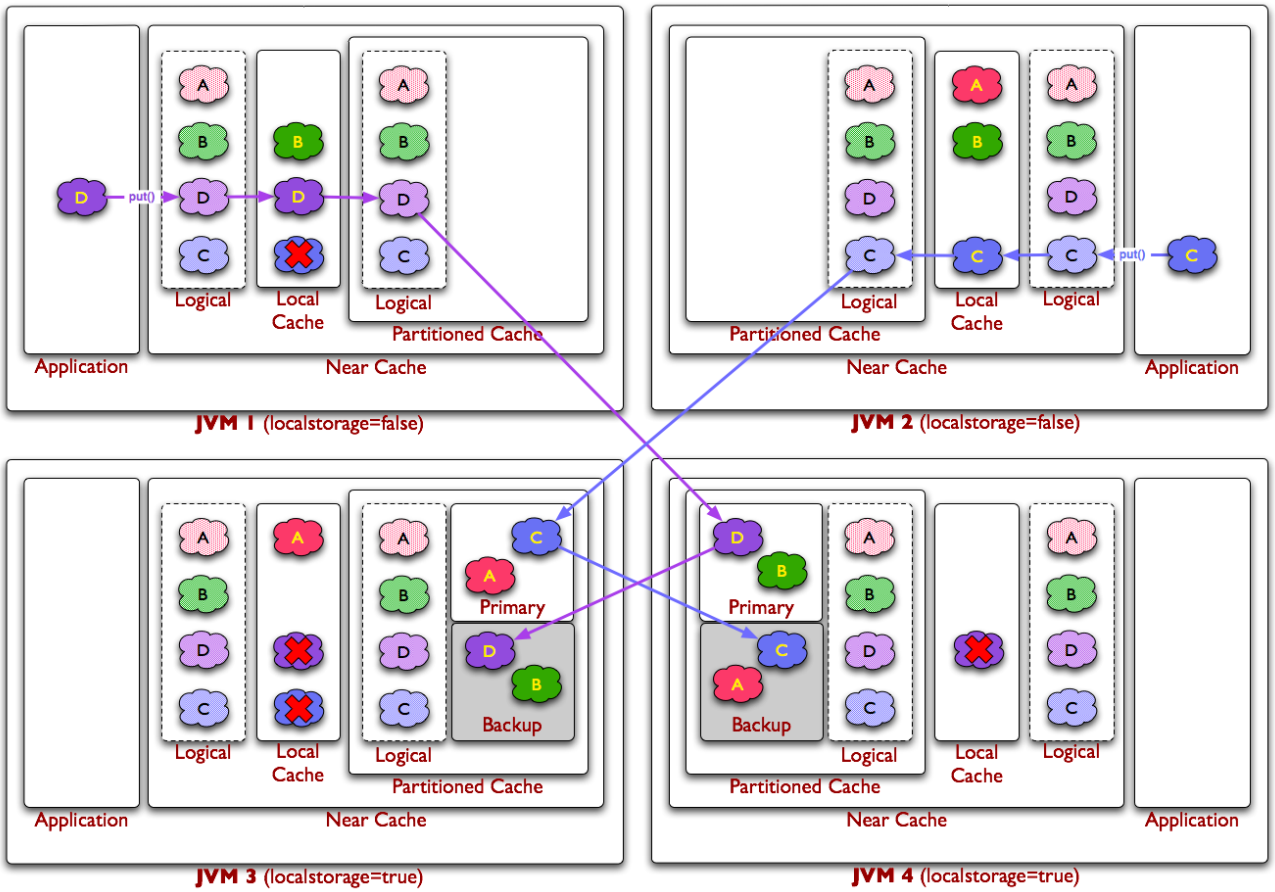
Region dzielony



Region dzielony



Region dzielony – optymalizacja



GemFire z punktu widzenia programisty…

- Bardzo bogate API
  - zdecydowaną większość konfiguracji można zmieniać w trakcie działania aplikacji
- Regiony implementują interfejs Map (Java)
  - dodatkowo umożliwiają dodanie listener’ów (słuchaczy?) aby być powiadamianym o zmianach
- Możemy redefiniować funkcję haszującą aby grupować określone dane ze sobą

Podsumowanie

- Istnieją zastosowania rozproszonych systemów danych, w których niedopuszczalna jest utrata spójności danych
- Twierdzenie CAP mówi, że coś stracić musimy
- Minimalizowanie tych strat jest zadaniem niebanalnym

Appendix 1

Strategie propagowania informacji

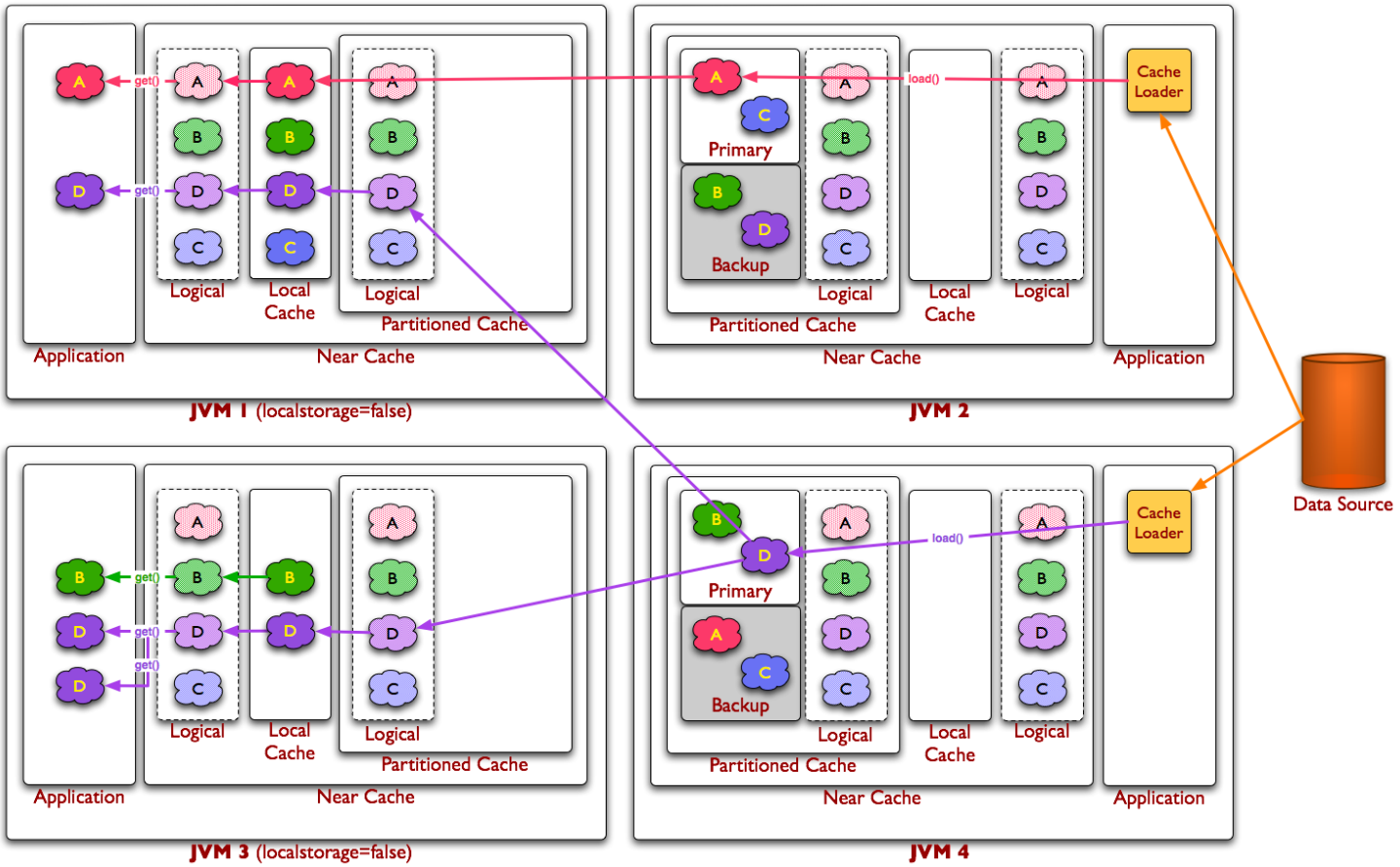
Strategie propagowania informacji

- local
  - brak rozpowszechniania – region jest widzialny tylko dla wewnętrznych wątków
- distributed-no-ack
  - zdarzenia są rozpowszechniane bez oczekiwania na potwierdzenie
- distributed-ack
  - zdarzenia są rozpowszechniane z oczekiwaniem na potwierdzenie
- global
  - zdarzenia są rozpowszechniane z blokowaniem całego regionu

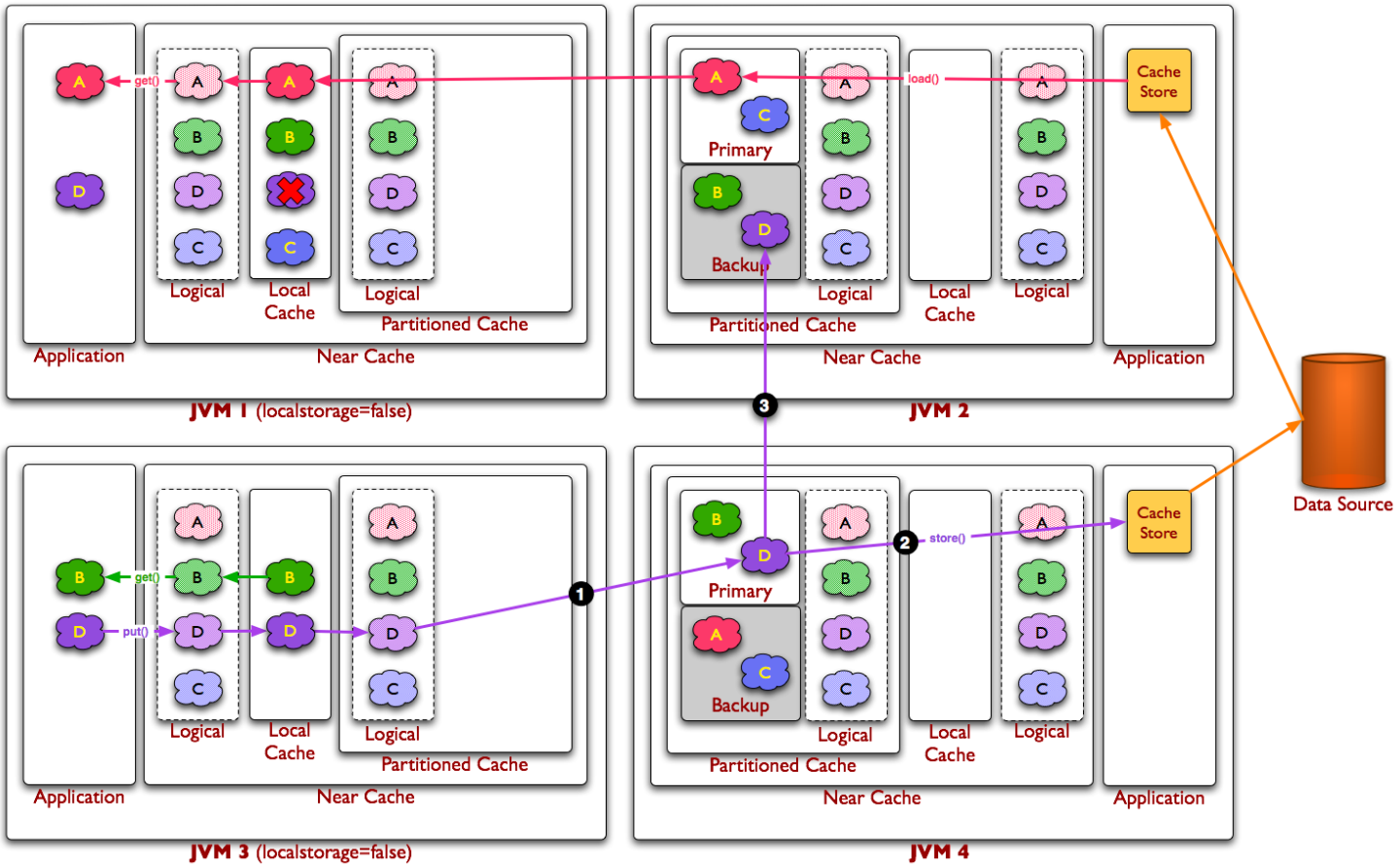
Appendix 2

Integracja z zewnętrznymi źródłami danych

Read Through



Write Through



Write Behind