Skalowanie aplikacji JPA

na przykładzie Oracle TopLink Grid

Waldek Kot

waldemar.kot@oracle.com

IT Project Management | PHP | .NET & C# | Java

March 26th, 2010 in Poznan, Poland

Agenda

- Krótko o JPA
- Skalowanie warstwy JPA
- Data Grid
- Demo
- Q&A

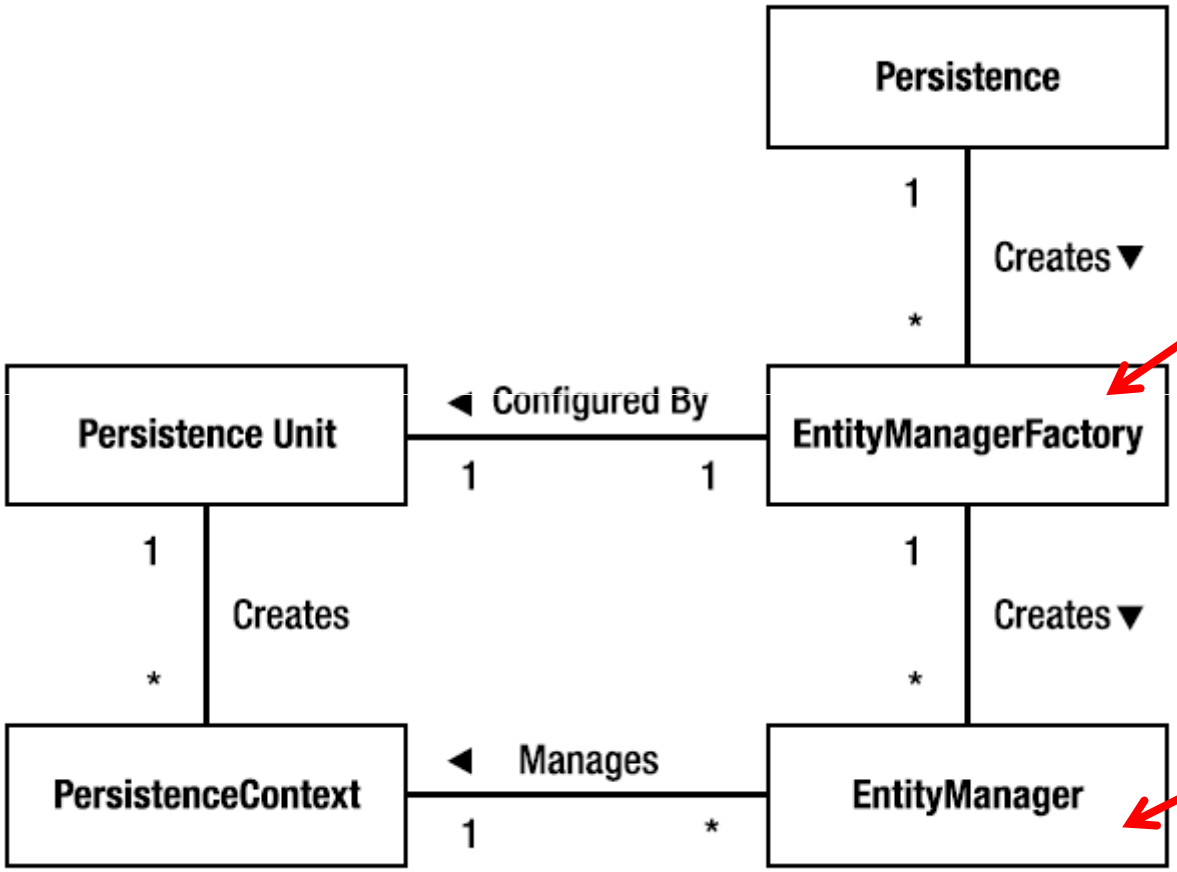
Java Persistence API (JPA)

- ORM – Object-Relational Mapping
- Wydajność i skalowalność
  - przepustowość, czas odpowiedzi, mniejsze obciążenie DB
  - możliwość bardziej inteligentnej współpracy z DB
  - cache (bufor)
  - locking
  - wykorzystanie cech DB (zachowując standardowe API)
  - kooperatywnie, nietransparentnie

Typowe podejścia do skalowania

- Cluster – grupa (współdziałających ze sobą) węzłów (kontenerów/dostawców)
  - Data grid – rozproszenie przechowywania i przetwarzania obiektów
- Optymalizacja komunikacji z bazą
  - Im rzadziej, tym lepiej
  - Im mniej danych transferowanych, tym lepiej
- Optymalizacja zapytań do bazy
  - Im krótsze i mniej obciążające, tym lepiej

Buforowanie w JPA



Level 2
„Shared cache”

Level 1
Cache „transakcyjny”

Konfiguracja cache/buforów

- Zależna od dostawcy JPA
- Zwykle możliwe
  - konfigurowanie które Entity będą buforowane
    - możliwość definiowania wskazówek (hint)
  - rozmiar bufora (liczba obiektów)
  - sposób lock'owania
  - sposób współpracy z Garbage Collection
    - typy referencji (weak/soft/hard/...)
- Parametry dobierane zależnie od sposobu używania Entity

Clustering

- Cache L1 zawsze pozostaje „prywatny”
- Cache L2 może być „współdzielony” między węzłami
  - węzły = niezależne JVM, potencjalnie uruchomione na oddzielnych maszynach/serwerach
  - koordynacja zawartości buforów L2 między węzłami
  - współdzielenie obiektów w cache pomiędzy węzłami
  - dedykowana warstwa buforowania
    - odciążenie węzłów od prac związanych z buforowaniem
    - możliwość zwiększenia pojemności cache
      - większy heap oraz mniejsza liczba kopii obiektów
    - przechowywanie w pamięci (ale w wysoce niezawodny sposób)
    - rozproszenie zapytań
    - read-through, write-through, refresh-ahead, **write-behind**
  - możliwość buforowania zarówno obiektów, jak i danych
    - jeśli obiekty, to unikamy (minimalizujemy) koszty mapowania, tworzenia obiektów i pomagamy GC

Przykład

- Oracle TopLink Grid
  - JPA + Data Grid
  - EclipseLink (Oracle TopLink) + Oracle Coherence
- Możliwe konfiguracje
  - **Cache**: rozproszony data grid (Coherence) jako L2 cache dla JPA (EclipseLink)
  - **Read**: dodatkowo, zapytania (odczyty) z JPA przechodzą najpierw przez data grid (Coherence), potem ewentualnie do DB
  - **Entity**: dodatkowo, operacje zapisywania obiektów z JPA przechodzą najpierw przez Coherence (potem ewentualnie do DB)
    - typowe konfiguracje: synchroniczny zapis do DB, asynchroniczny, opóźniony (write-behind), własne

DEMO 1
COHERENCE JAKO BUFOR L2 W JPA

DEMO 2
KONFIGURACJA READ

DEMO 2
KONFIGURACJA ENTITY (READ/WRITE) WRAZ Z WRITE-BEHIND

Podsumowanie

- warto połączyć zalety JPA i data grid
  - Standardowy interfejs (JPA)
  - Przezroczystość dla aplikacji
  - Selektywna konfiguracja buforowania
- warstwa data grid może być łatwo skalowana
  - zarówno **przechowywanie**, jak i **przetwarzanie** obiektów można rozproszyć
    - nawet na tysiące węzłów (maszyn)
- więcej informacji: http://otn.oracle.com

Pytania ?

Waldek Kot
Principal Systems Engineer, Central & Eastern Europe
Oracle Polska
Sienna 75, 00-833 Warszawa
Email: waldemar.kot@oracle.com
Mobile: +48 660 78 55 78