УДК 004.75

Ріпневський Олександр

ВИКОРИСТАННЯ СУБД ІЗ АРХІТЕКТУРОЮ IN-MEMORY DATA GRID

Анотація. В даній статті розглядається концепція надшвидкої обробки даних за допомогою СУБД з архітектурою In-Memory Data Grid. Ця технологія є більш ефективною для багатьох задач, які мають у своїй основі саме вирішення питань швидкодії та продуктивності. Також розглянуті деякі проблеми, що виникають при використанні цієї технології, та запропоновані шляхи їх розв’язання.

Ключові слова: СУБД, Великі Дані, швидкість обробки, система обробки даних у оперативній пам'яті, динамічне завантаження класів.

Аннотация. В данной статье рассматривается концепция сверхбыстрой обработки данных с помощью СУБД с архитектурой In-Memory Data Grid. Эта технология является более эффективной для многих задач, которые имеют в своей основе именно решение вопросов быстродействия и производительности. Также рассмотрены некоторые проблемы, возникающие при использовании этой технологии, и предложены пути их решения.

Ключевые слова: СУБД, Большие Данные, скорость обработки, система обработки данных в оперативной памяти, динамическая загрузка классов.

Summary. This article discusses the concept of ultra-fast data processing with the help of DBMS with the structure of In-Memory Data Grid. This technology is more effective for many tasks, which are based on the decision of the issues of speed and performance. Also, some arisen problems with the use of this technology are considered, and ways of their solution are suggested.

Keywords: DBMS, Big Data, processing speed, data processing system in operational memory, dynamic load of classes.

У ряді завдань може знадобитися дуже висока швидкість обробки даних. Наприклад, біржовим гравцям іноді потрібно миттєво прийняти рішення, ґрунтуючись на великій кількості даних про стан ринку, - за пару секунд ситуація вже може змінитися. Існує також цілий ряд завдань, коли рішення потрібно приймати в реальному часі, наприклад обробка біометричних даних, якінеобхіднообробляти в режимы реального часу, звіряючисьіз базою даних про зловмисників. Дуже велика швидкість надходження даних характерна для багатьох наукових завдань. Наприклад, проект по запуску гігантського радіотелескопа з сумарною площею антен 1 км^2, передбачає передачу сигналів з однієї антени зі швидкістю 160 Гбіт / с, що в 10 разів перевищує весь нинішній інтернет-трафік [1].

Багато бізнес-завдання і наукові експерименти вимагають спільної обробки даних різних форматів - це можуть бути табличні дані в СУБД, ієрархічні дані, текстові документи, відео, зображення, аудіофайли та т.д.

В останній час все більше компаній звертається до обробки даних безпосередньо в оперативній пам’яті. Це є наслідком стрімкого падіння вартості оперативної пам'яті (RAM), в результаті чого стає можливим зберігання всього

набору операційних даних в пам'яті, збільшуючи швидкість їх обробки більш ніж в 1000 разів, порівняно із обробкою даних на жорстких дисках. Інструментарій для побудови таких рішень надають продути In-Memory Compute Grid та In-Memory Data Grid.

По суті, In-Memory Data Grid - це розподілене сховище об'єктів. Воно забезпечує швидку обробку інформації і надвисоку доступність за допомогою розподіленого зберігання даних в оперативній пам'яті. Це радикально контрастує з підходом традиційних СУБД, які розроблені для зберігання даних на стабільних носіях. Оскільки процеси обробки даних у оперативній пам'яті протікають швидше, ніж звернення до файлової системи та зчитування інформації з неї, СУБД в пам'яті забезпечує на порядок більш високу продуктивність програмних додатків. Оскільки конструкція СУБД в пам'яті набагато простіша традиційних, то вони накладають набагато менші вимоги до об'єму пам'яті та характеристик центрального процесора.

Традиційні СУБД для прискорення роботи широко використовують кешування. Кешування це процес, в рамках якого традиційні СУБД зберігають часто використовувані записи в пам'яті для швидкого доступу до них. Однак, кешування прискорює лише процес пошуку необхідної інформації, а не її обробки. Відповідно, виграш у продуктивності істотно менший.

Є ряд функціональних особливостей, які відрізняють IMDG від інших продуктів, таких як IMDB, NoSql або NewSql бази даних. Одна з основних - по-справжньому масштабується секціонування даних (Data Partitioning) в кластері. IMDG по суті являє собою розподілену хеш-таблицю, де кожен ключ зберігається на строго певному сервері в кластері. Чим більше кластер, тим більше даних можна в ньому зберігати. Принципово важливим у цій архітектурі є те, що обробку даних слід проводити на тому ж сервері, де вони розташовані (локально), виключаючи (або зводячи до мінімуму) їх переміщення по кластеру. Фактично, при використанні добре спроектованого IMDG, переміщення даних буде повністю відсутнє за винятком випадків, коли в кластер додаються нові сервера або видаляються існуючі, змінюючи тим самим топологію кластера і розподіл даних в ньому.

Зберігання даних в IMDG – це лише половина функціоналу, необхідного для in-memory архітектури. Дані, що зберігаються в IMDG, необхідно обробляти паралельно та з високою швидкістю. Зазвичай in-memory архітектура секціонує дані в кластері за допомогою IMDG, а згодом виконуваний код відправляється саме на ті сервера, де знаходяться необхідні йому дані. Оскільки виконуваний код (обчислювальна задача) зазвичай є частиною обчислювальних кластерів (Computer Grids), і повинен бути правильно розвернутий (deployment), бути збалансованим при навантаженні (load-balancing), бути відмовостійким (fail-over), а також мати можливість запуску по розкладку (scheduling), інтеграція між Compute Grid та IMDG дуже важлива. Найбільш ефективною є система, в якій IMDG та Compute Grid є частинами одного і того же продукту та використовують одні, й ті ж сами API, це дозволяє добитися найбільшої продуктивності та надійності in-memory рішення.

Серед рішень, які на ринку використовуються найчастіше – рішення Oracle, IBM Cognos TM1, SAP HANA, Microsoft PowerPivot, QlikView і Pentaho

Business Analytics. Такі платформи добре використовувати при необхідності аналізу даних в режимі реального часу з урахуванням того, що в процесі аналізу дані можуть бути змінені в будь-який момент. Також такі рішення добре підходять у випадках, коли немає можливості створення багатовимірного сховища даних і потрібно проводити аналіз даних облікової системи без її модифікації. Системи пропонують різні способи горизонтального масштабування, як з використанням засобів самої платформи, так і з застосуванням додаткового ПЗ.

Важливим аспектом In-Memory Data Grid (IMDG) реалізації розподілених сховищ даних є динамічне завантаження класів.

Динамічне завантаження – це завантаження під час виконання певного процесу.

Будь який клас, що використовується в середовищі виконання, був так чи інакше туди завантажений будь-яким завантажником Java. Зазвичай класи завантажуються за необхідністю їх використання. Окрім цього базові класи завантажуються під час старту програми.

Існує два варіанти завантаження класів:

Статичне завантаження класів – це звичайне завантаження, що проводиться автоматично. При старті програми завантажник класів рекурсивно завантажує усі класи, що зустрічаються у програмі починаючи з основного класу. Об’єкти таких класів створюються стандартним способом за допомогою оператора new;

Динамічне завантаження класів відтворюється за допомогою використання Classloader. Динамічне завантаження має сенс, коли необхідно завантажити клас під час виконання програми, коли необхідно замінити клас, змінив якусь певну логіку, і коли при цьому перезапуск програми не є раціональним. На сьогодні склалась ситуація, коли жодна із систем, жодна з існуючих систем з архітектурою In-Memory Data Grid повністю не реалізує динамічне завантаження класів. Це призводить до необхідності перезавантаження системи, або окремої частини, де і буде використаний необхідний клас, і, відповідно, при необхідності внесення певних змін, неможливо буде це зробити без втрати даних. Крім того, такий підхід потребує часу на перезапуск.

Підхід динамічного завантаження дає змогу уникнути цих проблем. Перезавантаження будь-якого класу може бути ініційовано в будь-який момент часу, за умови що в цей час вже не відбувається перезавантаження цього класу. Під час операції завантаження тимчасово блокується можливість змінювати вже існуючі об’єкти старої версії класу, коли у випадку перезавантаження кластера доступ був би втрачений повністю.

Ідея динамічного завантаження класів, як і ідея звичайного завантаження, полягає у тому, що коли певні класи зазнали змін, що впливають на виконання програми, то необхідно перекомпілювати саме змінений файл для отримання актуальних результатів після проведених змін. Відмінністю є той факт, що при звичайному завантаженні є можливість призупинити сервіс або програму, що залежить від зміненого класу, перекомпілювати необхідний ресурс, а згодом

запустити всю систему з нуля, або перезапустити лише ту частину, на яку вплинули ці зміни.

Для забезпечення простоти та можливості використання інструменту динамічного завантаження в різних сервісах, було вирішено розробити модель, що повинна визначати зміни у файлах, що задіяні, а також самостійно завантажувати змінені класи. Маючи на увазі те, що перезавантаження має відбуватись в IMDG, було вирішено позиціонувати програму як доповнення до розподіленого сховища даних, що починає перезавантаження при надсиланні клієнтом класу.

Таким чином, приходимо до висновку, що сучасний стан функціонування СУБД із архітектурою In-Memory Data Grid потребує розробки алгоритму реалізації інструменту динамічного завантаження. На нашу думку цей алгоритм повинен складатися із 4-х етапів.

1) Ініціалізація: завантаження класу, конверторів, створення мапи зав'язків полів та конвертерів на основі описуючих фалів;
2) Підготовка: генерація конвертерів;
3) Обробка: створення нових об'єктів на основі старих;
4) Заміщення: заміщення старих даних новими.

Розробка вказаного алгоритму та програмного забезпечення на його основі значно підвищить швидкодію та надійність обробки Великих Даних у СУБД із архітектурою In-Memory Data Grid.

Література

1. Найдич А. Big Data: проблема, технология, рынок [Електронний ресурс]. Доступ: https://compress.ru/article.aspx?id=22725.

2. Обухов А. In-Memory. База данных в оперативной памяти [Електронний ресурс]. ECM Journal. Доступ: https://ecm-journal.ru/post/In-Memory-Baza-dannykh-v-operativnojj-pamjati.aspx

3. Underwood, Jen. Exploring 2017 gartner bi magic quadrant results [Електронний ресурс]. Доступ:http://www.jenunderwood.com/2017/02/22/2017-gartner-bi-magic-quadrant-results/

ДОВІДКА ПРО АВТОРІВ

Ріпневський Олександр – студент, кафедра СПіСКС, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського». Ripnevkij Oleksandr – student, Department of SP&SCS, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

E-mail: alexripnesky@gmail.com

Ріпневський Олександр

ВИКОРИСТАННЯ СУБД ІЗ АРХІТЕКТУРОЮ IN-MEMORY DATA GRID

Актуальність теми. В даній роботі розглянуті використання СУБД у розподілених сховищ даних. Описано основні функції та розглянуто механізми обробки даних в IMDG. Окремо розглянутий механізм завантаження класів до сховищ в оперативній пам'яті, та описано існуючі рішення його оптимізації.

Постановка проблеми. Відсутність добре інтерпретованого методу оптимізації механізму та створення СУБД до розподілених сховищ даних в оперативній пам'яті.

Постановка завдання. Завданням є описати використання баз даних і систем їх управління у використанні їх у розподілених сховищах даних та розробка алгоритму, що зможе використовувати можливості даного рішення максимально.

Викладення основного матеріалу. Проведено аналіз існуючих проблем у використанні звичайних баз даних у роботі систем розподілених даних. Описано можливі варіанти покращення їх роботи. Результати виявились добре інтерпретованими та змістовними.

Висновки. Проаналізовано параметри, проблематику та вирішення проблем, що мають місце під час роботи з розподіленими сховищами даних, які показали найкращі результати для генерації текстів. Було розроблено алгоритм, що дозволить використати можливості СУБД у розподілених сховищах в більшому степені.