2. *Цмоць І.Г.* Інформаційні технології та спеціалізовані засоби обробки сигналів і зображень у реальному часі. – Львів: УАД, 2005.- 227с.
3. *В.П. Гергель*. Высокопроизводительные вычисления для мнгопроцесорных многоядерных систем // Издательство Московского университета , 2010. – 544с.
4. *А.В. Борескови* др. Паралельные вычисления на GPU, Архитектрура и программная модель CUDA// Издательство Московского университета , 2012. – 336с.
5. *Грушицкий Р.И., Мурсаев А.Х., Угрюмов Е.П.* Проектирование систем на микросхемах программируемой логики. – СПб.: БХВ-Петербург, 2002. – 608с.
6. *С.Кун.* Матричные процессоры на СБИС:-М.:Мир,1991.-672 с.
7. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ: Пер. с англ. / Под ред. А. Шеня. – М.: МЦНМО: БИНОМ. Лаборатория знаний, 2004. – 960 с.
8. *Левитин Ананий.* Алгоритмы: введение в разработку и анализ. :Пер. с англ. - М.: Издательский дом «Вильямс», 2006. - 576с.
9. *Лорин Г.* Сортировка и системы сортировки. – М. : Мир. 1983. – 384 с.
10. *Мельничук А.С., Луценко С.П., Громовий Д.С., Трофимова К. В.* Аналіз методів сортування масиву чисел. Технологический аудит и резервы производства - №4/1(12), 2013. – С. 37-40.
11. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. — СПб: БХВ-Петербург, 2002. — 608 с.

Поступила 6.10.2016р.

УДК 004.75

І.Г. Цмоць, д.т.н., О.В. Скорохода, к.т.н., В.І. Роман
кафедра автоматизованих систем управління
Національний університет «Львівська політехніка», м. Львів

СХОВИЩА ДАНИХ БАГАТОРІВНЕВИХ СИСТЕМ УПРАВЛІННЯ ЕНЕРГОЕФЕКТИВНІСТЮ

Анотація. Досліджено та проаналізовано існуючі методи і засоби організації та розробки високонавантажених сховищ даних для організації доступу до великих даних (big data). Розглянуто лямбда-архітектуру, запропоновано технологічне рішення для реалізації сховища даних на її основі.

Abstract. Existing methods and means of organization and development of high-loaded warehouses for providing access to big data have been investigated and analysed. The lambda architecture has been considered, a technological solution for implementing a data warehouse based on it has been proposed.

Ключові слова*:* сховища даних, великі дані, багаторівневі системи управління, енергоефективність.

Keywords: data warehouses, big data, multi-level management system, energy efficiency.

Постановка проблеми

У наш час складні корпоративні системи повинні містити інтелектуальні компоненти, які здатні надавати рекомендації та об'єктивну інформацію у зручному для користувачів вигляді, або самостійно приймати певні управлінські рішення.

Інтелектуальні компоненти верхніх рівнів багаторівневих систем управління оперують історичною інформацією організації та використовуються для прийняття стратегічних рішень. Для цих компонент важливим є наявність ефективного сховища даних, яке керуватиме історичними даними організації.

Сховище даних повинне забезпечувати збір, попереднє опрацьовування, очищення, збереження і надання доступу до історичних даних для різних засобів опрацювання, зокрема, для інтелектуальних компонент.

Аналіз публікацій

У багаторівневих системах управління інформаційна інтеграція здійснюється шляхом формування єдиного інформаційного поля на основі об'єднання технологічного процесу збору, зберігання, передачі та опрацювання даних [1].

Базова структура багаторівневих систем управління може бути представлена у вигляді чотирирівневої структури [2-4]. Інформаційна інтеграція першого та другого рівня полягає у організації та опрацюванні історичних даних, необхідних для прийняття стратегічних рішень. Основними особливостями першого та другого рівня є: великий об'єм даних; різноманітність даних; суперечливість та неповнота даних; постійність і висока інтенсивність надходження вхідних даних. Ці особливості повинні бути враховані під час розробки сховища даних для системи.

Існує два підходи до розробки сховищ даних: так званий «класичний» підхід на основі реляційних баз даних та підхід на основі технологій BigData [5].

Класичний підхід успішно справляється з вирішенням задач інформаційної інтеграції вже десятиріччями [6]. Але останніми роками спостерігається інтенсивне збільшення об'ємів даних, їх різноманітності, неструктурованості та інтенсивності надходження у систему. Такі дані називають терміном BigData (великі дані). Класичні методи неефективні для опрацювання великих масивів даних, тому виникає потреба переглянути методи організації сховищ даних. Стек технологій BigData у своїй основі базується на кластерних технологіях і призначений для побудови сховищ даних що здатні ефективно збирати, накопичувати та опрацьовувати великі об'єми даних [6-7]. Цей стек досить новий у інформаційних технологіях. Сховища даних на основі BigData технологій зазвичай організовуються на основі лямбда архітектури, котра призначена для організації сховищ, які можуть надавати користувачам миттєву відповідь, яка міститиме найактуальніші дані [8].

Формулювання мети статті

Метою статті є запропонувати технологічне рішення для реалізації високоефективного розподіленого сховища даних на основі лямбда архітектури. Для цього потрібно виконати такі попередні кроки:

- проаналізувати існуючі методи та засоби для розробки високонавантажених сховищ даних, що здатні накопичувати та опрацьовувати великі масиви даних;
- розглянути стек BigData технологій, що призначений для побудови високонавантажених сховищ даних, розподілених на кластерах;
- розглянути лямбда архітектуру, яка призначена для розробки сховищ даних на основі технологій BigData, і при цьому забезпечує швидку відповідь сховища на запити користувача із врахуванням найновіших даних.

Основна частина

Класичний підхід до побудови сховищ даних базується на концепції розподілених вітрин даних, що місять агреговану часткову інформацію, що є актуальною тільки для конкретного підрозділу організації. Альтернативною концепцією реалізації сховищ даних є Data Lake концепція. Концепція передбачає побудову єдиного сховища, у якому «природнім чином» циркулюватимуть всі дані підприємства, як і сирі, так і очищені та агреговані. Власне ця концепція є одним з варіантів опрацювання великих даних. Єдиною на даних момент імплементацією цієї концепції є Apache Hadoop платформа.

Apache Hadoop – це програмна платформа для організації сховищ даних та опрацювання даних на розподілених обчислювальних кластерах. Особливістю Apache Hadoop є те, що його дизайн розроблений для роботи з великими об'ємами даних та інтенсивними обчисленнями над ними. Apache Hadoop має три основні складові: розподілену файлову систему (HDFS), менеджер ресурсів кластера (YARN) та обчислювальну платформу MapReduce. До недавнього часу Hadoop MapReduce був єдиним підходом до організації опрацювання даних у Hadoop системі, який має як і свої недоліки так і переваги, але з появою менеджера ресурсів кластера YARN з'явилася можливість створювати будь які класи розподілених робіт на Hadoop кластері абстрагуючись від ручного керування ресурсами кластера. Зокрема варто виділити технологію Apache Spark, що має можливість інтегруватися з Hadoop системою і вміє вирішувати широке коло задач, зокрема таких як машинне навчання, робота з графами та робота з даними за допомогою SQL запитів. Також Apache Spark здатна працювати у автономному режимі, має компоненти для організації та керування ресурсами кластеру і має широкі засоби інтеграції з найрізноманітнішими системами та технологіями.

Apache Hadoop платформа дає можливість організувати розподілене, високонадійне сховище даних та надає доступ до обчислювальних потужностей кластера. Крім цього платформа надає Hadoop MapReduce

компоненту для здійснення розподілених обчислень над даними у кластері. Окрім цих задач, для побудови ефективного розподіленого сховища даних потрібно забезпечити виконання ряду допоміжних задач. Для цих цілей навколо Apache Hadoop платформи був розроблений цілий набір технологій для спрощення розробки тих чи інших компонентів сховища даних. Технологія Apache HIVE дає можливість емулювати реляційну базу даних та SQL запити на базі HDFS та MapReduce або Spark SQL. Альтернативними технологіями для реалізації SQL на основі HDFS є Apache Impala та Apache Drill. Часто у зв'язці з Hadoop використовують NoSQL бази даних Cassandra та HBase для забезпечення швидкого пошуку агрегованих даних у кластерних сховищах, а також технологія Druid. Ці технології необхідні для забезпечення швидкої відповіді на запити користувачів та використовуються як компоненти лямбда архітектури. Для машинного навчання застосовуються технології Spark MLlib та Apache Mahout. Також існують засоби для керування та запуску ETL процесів: StreamSets, Oozie.

Окремим завданням є організація ETL процесів. Виділяють два види джерел даних: пакетні (batch) та потокові (streaming). Пакетні джерела передбачають запуск ETL процесів періодично для опрацювання цілого масиву даних. Для організації пакетної обробки даних використовуються Hadoop MapReduce, Spark Core, Apache Sqoop та інші. Потокові джерела характеризуються тим, що у них нові дані з'являються щосекунди і їх необхідно опрацювати та розмістити на кластері чи у тимчасовому сховищі. Основними технологіями потокової обробки даних є Spark Streaming, Apache Flume, Apache Samza, Apache Storm та інші.

Сховище даних повинне давати швидку відповідь на запити користувачів, при цьому можуть виникати ситуації, коли результати запитів повинні включати найсвіжішу інформацію, яка приходить до системи з потокових джерел. У таких ситуаціях виникає проблема швидкодії, оскільки для того, щоб враховувати найсвіжіші дані, необхідно виконати запит на всіх даних кластера у цей момент, що займає достатньо багато часу та не задовольняє потребу з швидкою відповіддю на запит. Є інший підхід, що полягає у переобчисленні запиту через певний інтервал часу та збереження його результатів у швидке сховище даних, зазвичай NoSQL базу. У цьому випадку користувач отримуватиме миттєву відповідь на його запит, але у цій відповіді не будуть враховані найсвіжіші дані. Для вирішення проблеми отримання швидкої відповіді на запит із врахуванням найновіших даних використовують лямбда архітектуру.

Лямбда архітектура складається з трьох рівнів. У пакетному рівні (batch layer) через певний інтервал часу виконуються обчислення підготовлених запитів користувачів на всіх даних сховища, результати зберігаються у обслуговуючий рівень (serving layer). Найсвіжіші дані, що прийшли у систему з потокових джерел та ще не попали на опрацювання у пакетному рівні, опрацьовуються швидкісним рівнем (speed layer). Користувач системи звертається із запитом до обслуговуючого рівня, котрий відповідає за

агрегацію результатів пакетного та швидкісного рівня. Таким чином повільні обчислення над всіма даними виконуються періодично у фоновому режимі і при запиті користувача їх результати вже наявні у системі, а найсвіжіші дані, котрі ще не були включені у ці результати, перебувають у швидкісному рівні та при запитах користувача можуть бути швидко опрацьовані, оскільки їх об'єми відносно не великі, або ж швидкісний рівень може неперервно переобчислювати результати запитів і зберігати їх у швидку базу даних у системі. Обслуговуючий рівень об'єднує результати запиту з обох рівнів та видає їх користувачеві. Такий підхід забезпечує швидку відповідь на запити із врахування у них найсвіжіших даних.

При виборі технологій, на основі яких будувати сховище даних, важливими їхніми характеристиками є актуальність та стабільність. Apache Hadoop платформа вже розвивається більше десяти років, містить багато допоміжних засобів та є достатньо надійною, тому на сьогодні є найкращим вибором у якості безплатного каркасу для сховища даних. Для здійснення пакетних обчислень себе зарекомендувала Apache Spark, як стабільна технологія, що динамічно розвивається та показує кращу швидкодію ніж Hadoop MapReduce для більшості завдань. Крім цього, Apache Spark містить компонент потокової обробки даних Spark Streaming, рушій для опрацювання SQL запитів Spark SQL та модулі машинного навчання і роботи з графами. Важливою перевагою є не тільки широкий діапазон задач, які здатен вирішувати Apache Spark, але і цілісність всіх його компонентів та спільне ядро, що значно полегшує розробку, підтримку та інтеграцію його компонентів. При виборі лямбда архітектури важливу роль відіграє черга повідомлень, котра виступає буфером між вхідними джерелами даних та пакетним і швидкісним рівнями архітектури. Для створення такої черги повідомлень для Big Data сховищ даних себе зарекомендувала технологія Apache Kafka, як надійний, розподілений та місткий буфер повідомлень. При виборі компоненти для обслуговуючого рівня можна розглядати одну з NoSQL баз даних – Cassandra або HBase, та технологію Druid. Перевага технології Druid полягає у тому, що вона спеціально розроблена для збереження, опрацювання та агрегації результатів пакетного та швидкісного рівнів, тому вона ідеально підходить для лямбда архітектури.

Описана архітектура сховища даних базується на стабільних, функціональних, гнучких та легко масштабованих на кластері компонентах, що забезпечує побудову ефективного сховища даних на основі лямбда архітектури. Описана архітектура зображена на рис. 1.

Висновки

Для формування інформаційного середовища організації, яка оперує великими масивами даних, які є важливими для прийняття стратегічних рішень, використовується технологічний стек BigData, який забезпечує створення високоефективних розподілених сховищ даних.

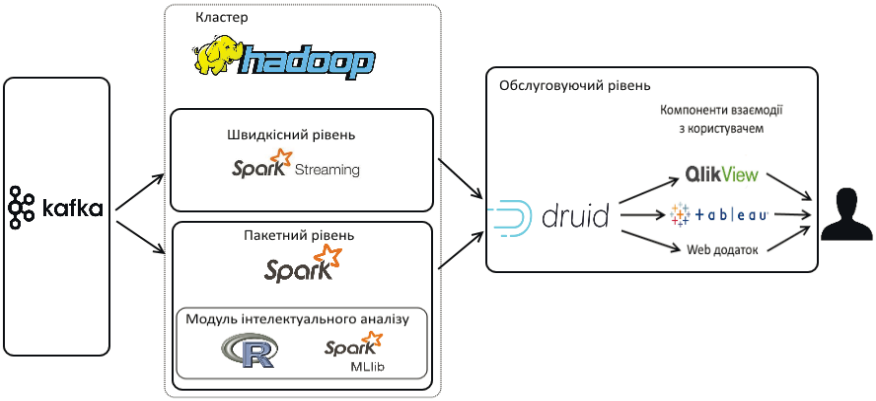

Рис. 1. Архітектура сховища даних на основі Big Data технологій та лямбда архітектури

У статті проведено аналіз існуючих технологій для організації сховищ даних, зокрема розглянутий стек BigData технологій. Розглянута лямбда архітектура, що широко використовується у BigData рішеннях, та запропоноване технологічне рішення для реалізації сховища даних на її основі.

1. Інтелектуальні компоненти інтегрованих автоматизованих систем управління: монографія / Медиковський М.О., Ткаченко Р.О., Цмоць І.Г., Цимбал Ю.В., Дорошенко А.В., Скорохода О.В. – Львів: Видавництво Львівської політехніки, 2015. – 280 с.
2. *Barry, W. L., et al.* "Manufacturing planning and control for supply chain management." (2010).
3. *O'Leary, Daniel E.* Enterprise resource planning systems: systems, life cycle, electronic commerce, and risk. Cambridge university press, 2000.
4. *Chiu, Yu Hsien, et al.* "Enterprise resource planning." (2014).
5. Big Data vs. the Data Warehouse [Електронний ресурс]. – Режим доступу: http://www.information-management.com/blogs/blog/big-data-vs-the-data-warehouse-10025458-1.html
6. Inmon-vs-Kimball-Which-approach-is-suitable-for-your-data-warehouse [Електронний ресурс]. – Режим доступу: http://www.computerweekly.com/tip/ Inmon-vs-Kimball-Which-approach-is-suitable-for-your-data-warehouse
7. Big Data & Analytics Reference Architecture [Електронний ресурс]. – Режим доступу: http://www.oracle.com/technetwork/ topics/entarch/oracle-wp-big-data-refarch-2019930.pdf
8. Big Data for Data Warehousing [Електронний ресурс]. – Режим доступу: http://www.tcs.com/SiteCollectionDocuments/White-Papers/BFS-Whitepaper-Big-Data-Warehousing-0313-1.pdf
9. Lambda Architecture [Електронний ресурс]. – Режим доступу: http://lambda-architecture.net/

Поступила 13.10.2016р.