КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

Y. Tupalo

ARCHITECTURE OF EXPERT SUPPORT AND DECISION MAKING SYSTEM

The main objective of this paper is to show that the development of technical tools, mobile devices can be used not only for communication but also as a powerful platform to calculate complex mathematical problems.
Key words: СППР, SMP, MPP, NUMA, TEE, REE, TCB, DI, IoC.

Главная цель данной работы показать, что с развитием технических средств, мобильные устройства можно использовать не только для связи, а также как платформу для вычисления сложных математических задач.
Ключевые слова: СППР, SMP, MPP, NUMA, TEE, REE, TCB, DI, IoC.

Головна мета даної роботи показати, що з розвитком технічних засобів, мобільні пристрої можна використовувати не лише для зв'язку, а також як потужну платформу для обчислення складних математичних задач.
Ключові слова: СППР, SMP, MPP, NUMA, TEE, REE, TCB, DI, IoC.

УДК 004.4

Я.О. ТУПАЛО

АРХІТЕКТУРА СИСТЕМИ ПІДТРИМКИ ТА ПРИЙНЯТТЯ РІШЕННЯ

Вступ. Сьогодні основним чинником створення тривалої конкурентної переваги і зростання інвестиційної привабливості компанії стають оптимальні стратегії управління бізнесом. Ефективне управління – це такий же ресурс, як гроші або матеріальні цінності.

Підвищення ефективності управлінської діяльності стає одним з напрямків вдосконалення діяльності підприємства в цілому. Найбільш очевидним засобом підвищення ефективності трудового процесу є його автоматизація. Але те, що можливе для строго формалізованого виробничого процесу, становиться складним для такої сфери, як управління. Особливо коли приймається не очевидне рішення, від якого залежить подальша доля підприємства. Рішення, яке приймає вище керівництво підприємства може вмістити в собі безліч факторів, які не очевидні навіть для досвідчених фахівців. Одним з можливих підходів є впровадження систем підтримки прийняття рішень (СППР). Тому обрана тема є актуальною і потребує подальшого дослідження.

Технологічні проекти зазвичай – найскладніші для оцінювання. Особливо це стосується СППР-проектів [1]. Процес розроблення таких великомасштабних проектів інформаційних систем є надзвичайно активним. Оцінювання СППР-проектів треба здійснювати на всіх етапах життєвого циклу.

На основі викладеного можна сформулювати ціль статті, яка полягає у поєднані різноманітних технологій для побудови маштабованої архітектури СППР.

Керівництво великих підприємств, організацій, компаній тощо постійно потребує до-

стовірної інформації з різних аспектів бізнес-процесів для підтримки прийняття рішень. Від отриманої інформації залежить якість управління, ефективність планування діяльності, виживання в умовах жорсткої конкурентної боротьби. При цьому критично важливими є наочність форм подання інформації, оперативність отримання різних видів звітності, можливість аналізу поточних та історичних даних. Системи, що забезпечують такі можливості відомі під назвою – СППР. Вони з успіхом застосовуються в різних галузях електронного документообігу: телекомунікаціях, фінансовій сфері, торгівлі, промисловості, медицині тощо.

За допомогою СППР може проводитись вибір рішень у певних неструктурованих і слабо структурованих задач, у тому числі й тих, що мають багато критеріїв. Тому сучасні системи підтримки прийняття, що виникли в результаті злиття керівницьких інформаційних систем і систем керування базами даних, це системи, що максимально пристосовані до розв'язання задач щоденної керівницької діяльності і є інструментом, покликаним надати допомогу тим, хто вирішує.

Загальна частина. Потенціал IT-індустрії на сьогоднішній день зосереджений у більшій мірі на створення і розвиток мобільних гаджетів та програм для них. За даними компанії J'son&Partners Consulting зростання об'єму світового ринку мобільних додатків зростає у геометричній прогресії. Найбільш розповсюджені групи мобільних додатків за типом є ігри, навігація в Інтернеті, соціальний бізнес, пошукові сервіси, поштові сервіси, музичні сервіси, мобільні платіжні системи, медіа і бізнес аналітика. Остання група розвивається дуже швидко, тому що в бізнесі швидкість прийняття рішення виступає на першому місці. Тому автор даної статті в подальшому розгляне власну архітектуру побудови мобільної СППР. Ми бачимо, що кожен день у сервісах мобільних додатків з'являються нові бізнес додатки, що говорить про ріст популярності мобільних рішень серед бізнес-користувачів.

З усього різнобарв'я додатків мобільні технології доступу до ділової інформації, на даний момент, є найбільш незаповненим сегмент розвитку IT-індустрії, і процес пошуку нових можливостей для використання мобільних гаджетів знаходиться в стадії стрімкого розвитку.

Дослідження, яке проводило IBM Tech Trends Report [2], показує, що майже 70 % організацій по всьому світу збільшують інвестиції у мобільні технології, більше половини збільшать витрати на бізнес аналітику. На думку IBM, чотири галузі IT-індустрії, мобільні технології, бізнес аналітика, "хмарні" обчислення і соціальний бізнес стрімко змінюють потреби і процеси функціонування підприємства. Мобільні рішення будуть більш широко використовуватися серед менеджменту не тільки великих і середніх компаніях, але і компаніями малого бізнесу.

Мобільні технології доступу до ділової інформації на сьогодні замінюють комп'ютерні рішення, відкриваючи цим нову еру мобільного керування бізнесом, мобільні СППР.

На думку автора, забезпечення мобільними рішеннями менеджменту підприємств є перспективною задачею. На сьогоднішній день, як показують соціаль-

ні опитування, керівники підприємств майже не використовують новітні інформаційні рішення для керування бізнесом. Запропоновані на даний момент рішення, по-перше, призначені для стаціонарних комп'ютерів, по-друге, пригодні для введення бухгалтерського обліку і фінансового аналізу вже сформовані іншими спеціалістами.

Вищенаведені факти говорять про необхідність побудови масштабованої і гнучкої архітектури мобільної СППР. На рис. 1 показано трирівнева архітектура мобільної СППР.

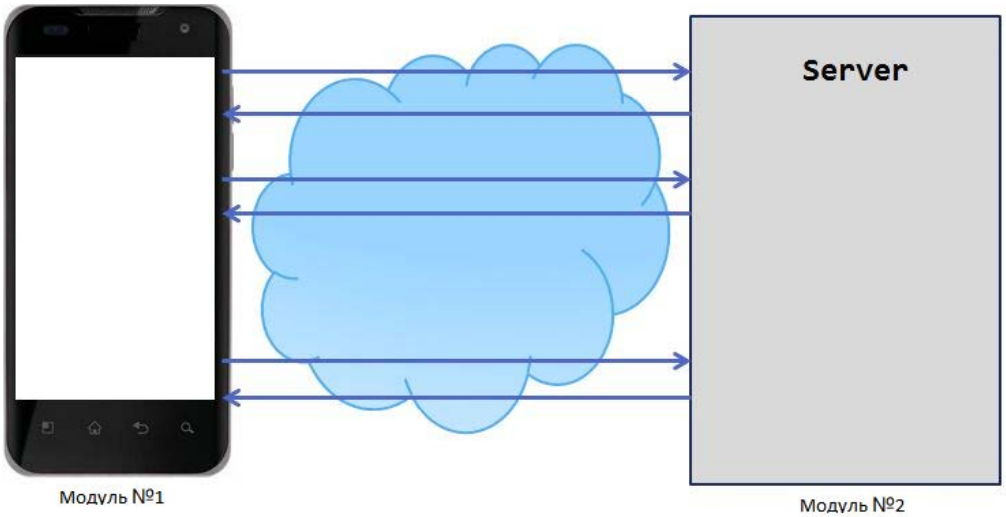

РИС. 1. Трирівнева архітектура мобільної СППР

Як правило комп'ютери і програми, що входять до складу інформаційної системи, не є рівноправними. Деякі з них володіють ресурсами, інші мають можливість звертатися до цих ресурсів. Комп'ютер, керуючий ресурсом, називають сервером цього ресурсу. Клієнт і сервер будь-якого ресурсу можуть знаходиться як на одному комп'ютері, так і на різних комп'ютерах, пов'язаних мережею.

В рамках багаторівневого подання обчислювальних систем можна виділити три групи функцій, орієнтованих на рішення різних підзадач:

- функції введення і відображення даних;
- прикладні функції, характерні для даної предметної області;
- функції керування ресурсами.

Сервер і мобільний телефон поєднуються через API (application programming interface) – набір готових класів, процедур, функцій, структур і констант для використання у зовнішніх програмних продуктах. Використовується програміста-

ми при написанні всіляких додатків. API визначає функціональність, яку надає програма (модуль, бібліотека), при цьому API дозволяє абстрагуватися від того, як саме ця функціональність реалізована. Якщо програму (модуль, бібліотеку) розглядати як чорний ящик, то API – це безліч «ручок», які доступні користувачеві даного ящика і які він може крутити і смикати. Програмні компоненти взаємодіють один з одним за допомогою API. При цьому зазвичай компоненти утворюють ієрархію – високорівневі компоненти використовують API низькорівневих компонентів, а ті, в свою чергу, використовують API ще більш низькорівневих компонентів.

Надалі сервер будемо вважати обчислювальною системою. Сучасні обчислювальні системи будують за трьома архітектурами: SMP архітектура, MPP архітектура і NUMA архітектура.

SMP (symmetric multiprocessing) – симетрична багатопроцесорна архітектура. Головною особливістю систем з архітектурою SMP є наявність загальної фізичної пам'яті, що розділяється всіма процесорами (рис. 2).

РИС. 2. Архітектура SMP

Пам'ять служить, зокрема, для передачі повідомлень між процесорами, при цьому всі обчислювальні пристрої при зверненні до неї мають рівні права й одну і ту ж адресацію для всіх елементів пам'яті. Тому SMP-архітектура називається симетричною. Остання обставина дозволяє дуже ефективно обмінюватися даними з іншими обчислювальними пристроями. SMP-система будується на основі високошвидкісної системної шини (SGI PowerPath, Sun Gigaplane, DEC TurboLaser), до слотів якої підключаються функціональні блоки типів: процесори (ЦП), підсистема введення/виводу (I/O) і т. п. Для під'єднання до модулів I/O використовуються вже повільніші шини (PCI, VME64). Найбільш відомими SMP-системами є SMP-сервер і робочі станції на базі процесорів Intel (IBM, HP, Compaq, Dell, ALR, Unisys, DG, Fujitsu й ін.). Вся система працює під управлінням єдиної ОС (зазвичай UNIX-подібної, але для Intel-платформ підтримується Windows NT). ОС автоматично (в процесі роботи) розподіляє процеси по процесорах, але іноді можлива і явна прив'язка.

Недоліки:

- системи із загальною пам'яттю погано масштабуються.

Цей істотний недолік SMP-систем не дозволяє вважати їх по-справжньому перспективними. Причиною поганої масштабованості є те, що в даний момент шина здатна обробляти тільки одну транзакцію, внаслідок чого виникають проблеми вирішення конфліктів при одночасному зверненні декількох процесорів до одних і тих областей загальної фізичної пам'яті. Обчислювальні елементи починають один одному заважати. Коли відбудеться такий конфлікт, залежить від швидкості зв'язку та від кількості обчислювальних елементів. В даний час конфлікти можуть відбуватися при наявності 8–24 процесорів. Крім того, системна шина має обмежену (хоч і високу) пропускну здатність і обмежене число слотів. Усе це очевидно перешкоджає збільшенню продуктивності при збільшенні числа процесорів і кількості підключених користувачів. У реальних системах можна задіяти не більше 32 процесорів. Для побудови масштабованих систем на базі SMP використовуються кластерні або NUMA-архітектури. При роботі з SMP-системами використовують так звану парадигму програмування з пам'яттю (shared memory paradigm).

MPP (massive parallel processing) – масивно-паралельна архітектура. Головна особливість такої архітектури полягає у тому, що пам'ять фізично розділена. В цьому випадку система будується з окремих модулів, що містять процесор, локальний банк операційної пам'яті (ОП), комунікаційні процесори або мережеві адаптери, іноді – жорсткі диски і/або інші пристрої введення/виводу. За суттю, такі модулі представляють собою повнофункціональні комп'ютери (рис. 3). Доступ до банку ОП з даного модуля мають тільки процесори (ЦП) з цього ж модуля. Модулі з'єднуються спеціальними комунікаційними каналами. Користувач може визначити логічний номер процесора, до якого він підключений, і організувати обмін повідомленнями з іншими процесорами. Використовуються два варіанти роботи операційної системи (ОС) на машинах MPP-архітектури. В одному повноцінна операційна система (ОС) працює тільки на керуючу машину (front-end), на кожному окремому модулі функціонує сильно урізаний варіант ОС, що забезпечує роботу тільки розташованої у ньому гілки паралельного застосування. У другому варіанті на кожному модулі працює повноцінна UNIX-подібна ОС, що встановлюється окремо.

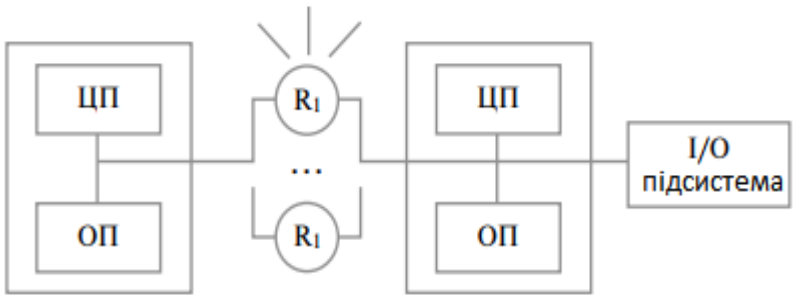

РИС. 3. Схематичний вид архітектури з роздільною пам'яттю

Головна перевага систем з роздільною пам'яттю – це хороша масштабованість: на відміну від SMP-систем, в машинах з роздільною пам'яттю кожен процесор має доступ тільки до своєї локальної пам'яті, у зв'язку з чим не виникає необхідності у потактовій синхронізації процесорів. Практично всі рекорди за продуктивністю на сьогодні встановлюються на машинах саме такої архітектури, які перебувають з декількох тисяч процесорів (ASCI Red, ASCI Blue Pacific).

NUMA архітектура (Nonuniform memory access) – неоднорідний доступ до пам'яті. Гібридна архітектура поєднує переваги систем із спільною пам'яттю і відносну дешевизну систем з роздільною пам'яттю. Суть цієї архітектури – в особливій організації пам'яті, а саме: пам'ять фізично розподілена за різними частинами системи, але логічно вона є спільною, так що користувач бачить єдиний адресний простір. Система побудована з однорідних базових модулів (плат), що складаються з невеликого числа процесорів і блоку пам'яті. Модулі об'єднані за допомогою високошвидкісного комутатора. Підтримується єдиний адресний простір, апаратно підтримується доступ до віддаленої пам'яті, тобто до пам'яті інших модулів. При цьому доступ до локальної пам'яті здійснюється в кілька разів швидше, ніж до віддаленої. За суттю архітектура NUMA – це MPP (масивно-паралельна) архітектура, де як окремі обчислювальні елементи беруться вузли симетричної багатопроцесорної архітектури (СБП). Доступ до пам'яті та обмін даними всередині одного SMP-вузла здійснюється через локальну пам'ять вузла і відбувається дуже швидко, а до процесорів іншого SMP-вузла теж є доступ, але більш повільний і через більш складну систему адресації.

Структурна схема комп'ютера з гібридною мережею: чотири процесори зв'язуються між собою за допомогою кросбара в рамках одного SMP-вузла. Вузли зв'язані мережею типу «метелик» (Butterfly) (рис. 4).

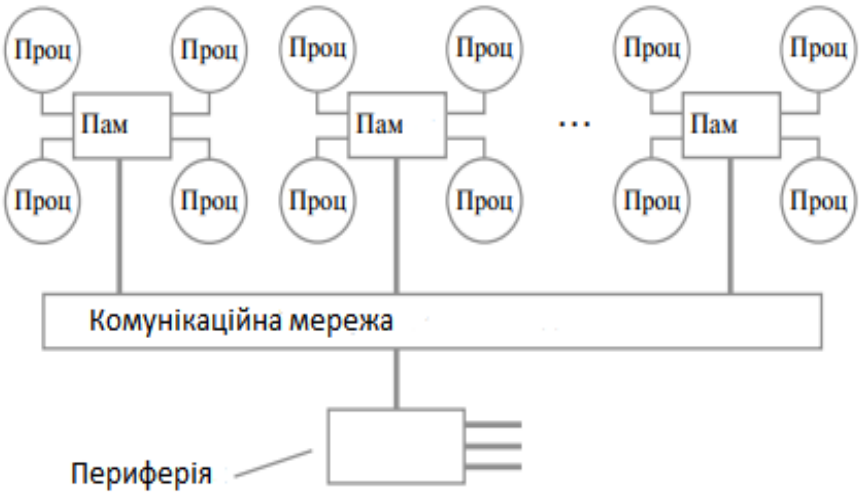

РИС. 4. Структурна схема комп'ютера з гібридною мережею

Останнім часом все більше зазнає критика небезпечності зберігання корпоративної інформації на стороні клієнта (в нашому випадку на мобільному телефоні). На дану критику автор пропонує застосовувати так звану Trusted Execution Environment (TEE) технологію.

Безпечне середовище виконання TEE технології [3] характеризується захищеністю, контролем цілісності та наявністю власної оперативної пам'яті і простору зберігання. Вона ізольована від звичайного «функціонально багатого середовища виконання» Rich Execution Environment (REE), в якій працюють операційна система і додатки мобільного пристрою. Користуючись можливостями TEE, розробники можуть створювати додатки і сервіси REE, які залишаються захищеними навіть при компрометації операційної системи – ризиковані операції ізольовані від REE, а конфіденційні дані (наприклад, криптографічні ключі) ніколи не покидають TEE.

Мобільні пристрої мають довірену обчислювальну базу Trusted Computing Base (TCB), що складається з апаратних і мікропрограмних компонентів, які повинні бути абсолютно надійними. Цілісність мобільної платформи перевіряється в рамках процесу захищеного або аутентифікованого завантаження.

Механізм захищеного зберігання може давати REE доступ до певного набору криптографічних алгоритмів, гарантуючи, що ключі шифрування ніколи не покинуть апаратну TCB. Для багатьох сервісів досить стандартних криптографічних операцій, але деяким додаткам REE може знадобитися виконання спеціальних алгоритмів в ізоляції від мобільної ОС і решти REE. Приклад – пропрієтарні алгоритми створення разових паролів для онлайн-банкінгу.

Для того, щоб у захищеній енергозалежній пам'яті можна було ізольовано виконувати довільний код, TEE має надавати відповідний інтерфейс програмування. Сам код для цього має бути підписаний сертифікатом, що містить його хеш і завіреним з використанням кореня довіри пристрою. Такий сертифікат дає право на виконання коду в TEE і доступ до ключа пристрою. Управління доступом до ключа здійснюється TCB з урахуванням інформації про стан платформи, збереженої при аутентифікованому завантаженні в енергозалежній цілісно захищеній пам'яті.

Деякі мобільні платформи надають API для доступу до апаратних засобів безпеки. Платформа Java ME [4], широко застосовувалася в мобільних телефонах, реалізує JSR 177 – інтерфейс для зв'язку з захищеними елементами і виклику функцій шифрування. JSR 177 може підтримуватися захищеним елементом мобільного телефону, наприклад SIM-картою. З недавнього часу в Android з'явився API для роботи з апаратно реалізованими функціями шифрування, що відповідає стандарту PKCS # 11, а в iOS аналогічні функції надає пропрієтарні API. Всі ці API високого рівня розроблялися з розрахунком на використання з апаратними модулями безпеки, захисними токенами і смарт-карт. Вони дозволяють генерувати апаратно захищені ключі і виконувати стандартні криптографічні операції на кшталт зашифрований текст.

Сьогодні навіть у середніх компаніях в день генерується дуже велика кількість даних, яку вже не можливо обробити в таблицях Excel, тому без алгоритмів великих даних не обійтися, особливо в розробці мобільних СППР.

Великі дані (англ. Big Data) – серія підходів, інструментів і методів обробки структурованих і неструктурованих даних величезних обсягів і значного різноманіття для отримання зрозумілих людиною результатів, ефективних в умовах безперервного приросту, розподілу за численними вузлами обчислювальної мережі, що сформувалися в кінці 2000-х років, альтернативних традиційним системам управління базами даних і рішень класу Business Intelligence.

Таким чином під Big Data я буду розуміти не якийсь конкретний обсяг даних і навіть не самі дані, а методи їх обробки, які дозволяють паралельно обробляти інформацію. Ці методи можна застосувати як до величезних масивів даних (таким як зміст усіх сторінок в інтернеті), так і до маленьких (таким як вміст цієї статті).

Виходячи з визначення Big Data, можна сформулювати основні принципи роботи з такими даними:

- горизонтальна масштабованість. Оскільки даних може бути як завгодно багато – будь-яка система, яка має на увазі обробку великих даних, має бути розширювана. У 2 рази зріс обсяг даних – у 2 рази збільшили кількість заліза в кластері і все продовжило працювати;

- стабільність. Принцип горизонтальної масштабованості має на увазі, що машин у кластері може бути багато. Наприклад, Hadoop-кластер Yahoo має більш ніж 42000 машин. Це означає, що частина цих машин буде гарантовано виходити з ладу. Методи роботи з великими даними мають враховувати можливість таких збоїв і переживати їх без будь-яких значущих наслідків.

- локальність даних. У великих розподілених системах дані розподілені за великою кількістю машин. Якщо дані фізично знаходяться на одному сервері, а обробляються на іншому – витрати на передачу даних можуть перевищити витрати на саму обробку. Тому одним з найважливіших принципів проектування Big Data-рішень – це принцип локальності даних, за можливістю обробляються дані на тій же машині, на якій їх зберігаємо.

Всі сучасні засоби роботи з великими даними так чи інакше відповідають цим трьом принципам. Для того, щоб їх дотримуватися – необхідно розробляти якісь методи, способи і парадигми розробки засобів обробки даних.

В архітектурі мобільної СППС автор пропонує використовувати алгоритм MapReduce. Алгоритм MapReduce – це модель розподіленої обробки даних, запропонована компанією Google для обробки великих обсягів даних на комп'ютерних кластерах (рис. 5).

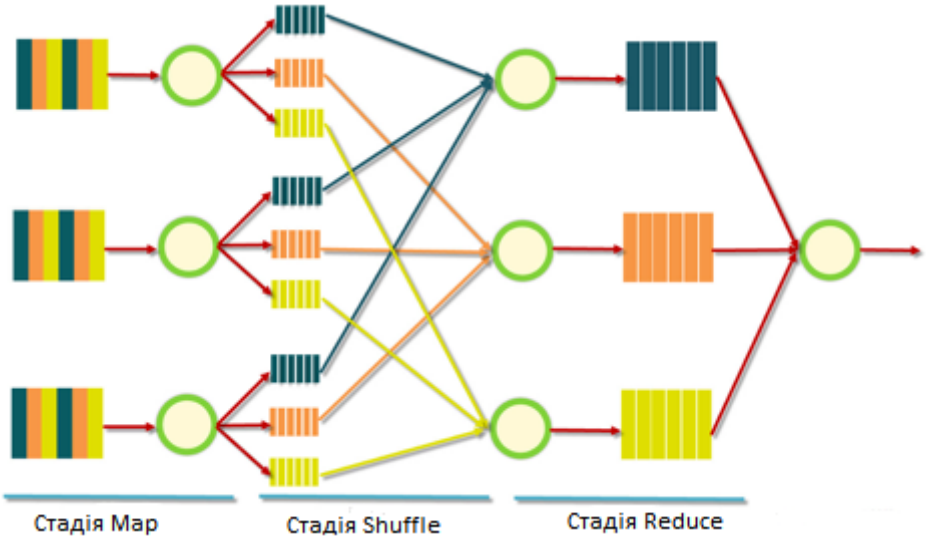

РИС. 5. MapReduce

Стадія Map. На цій стадії дані змінюються за допомогою функції map (), яку визначає користувач. Робота цієї стадії полягає у передобробці і фільтрації даних. Робота дуже схожа на операцію map в функціональних мовах програмування – призначена для користувача функція застосовується до кожного вхідного відрізку. Функція map () застосовується до вхідного запису і видає безліч пар ключ-значення. Безліч – тобто може видати тільки один запис, може не видати нічого, а може видати кілька пар ключ-значення. Що буде знаходиться у ключі і в значенні – вирішувати користувачу, але ключ – дуже важлива річ, так як дані з одним ключем у майбутньому потраплять в один екземпляр функції reduce.

Стадія Shuffle. Проходить непомітно для користувача. У цій стадії висновок функції map «розбирається по кошиках» – кожна корзина відповідає одному ключу виведення стадії map. Надалі ці кошики послужать входом для reduce.

Стадія Reduce. Кожен «кошик» із значеннями, який сформований на стадії shuffle, потрапляє на вхід функції reduce (). Функція reduce задається користувачем і обчислює фінальний результат для окремого «кошика». Безліч всіх значень, повернутих функцією reduce (), є фінальним результатом MapReduce-завдання.

За приклад можна взяти модуль зберігання документів. Завдання формулюється так: є великий корпус документів. Завдання – для кожного слова, яке хоча б один раз зустрічається в корпусі, порахувати сумарну кількість разів, яке воно зустрілося в корпусі.

Рішення. Раз маємо великий корпус документів – нехай один документ буде одним вхідним записом для MapRreduce-завдання. У MapReduce ми можемо тільки задавати призначені для користувача функції, що ми і зробимо (будемо використовувати python-like псевдокод):

```
def map(doc): for word in doc: yield word, 1
def reduce(word, values): yield word, sum(values)
```

Функція map перетворює вхідний документ в набір пар (слово, 1), shuffle прозоро для нас робить з цього в пари (слово, [1,1,1,1,1,1]), reduce підсумовує ці одинички, повертаючи фінальну відповідь для слова.

Висновки. Таким чином на даному етапі розвитку технологій необхідність розробки інформаційної системи для підтримки і прийняття рішення для малого і середнього бізнесу з використанням мобільних технологій, хмарних обчислень і бізнес аналітики, з елементами сервісів у сторону розвитку інтелектуальних і експертних систем на базі мультиплатформенних рішень залишилась безсуперечною. В даній статті автор запропонував стек технологій і методології для розробки масштабованої і гнучкої до потреб малого і середнього бізнесу, інформаційну систему підтримки і прийняття рішення на клієнтській стороні використовується мобільний пристрій а всі складні операції з обчисленнями переносяться на сторону серверу.

1. Урінцов А.І., Дік В.В. системи підтримки прийняття рішень. М.: МЄСИ, 2008.
2. The 2012 IBM Teach Trends Report // [електроний ресурс] http:// citizenibm. com/ 2012/12/analytics-education-must-center-on-learners.html
3. Kostiainen K. et al. Old, New, Borrowed, Blue – A Perspective on the Evolution of Mobile Platform Security Architectures, Proc. 1st ACM Conf. Data and Application Security and Privacy(CODASPY). 2011. P. 13–24.
4. Srage J. and Azema J. M-Shield Mobile Security Technology, white paper. Texas Instruments. 2005.

Одержано 21.09.2017