Порівняльний аналіз технології паралельного обчислення великих масивів даних MapReduce

Т.В.Борис, М.О.Алєксєєв

Національний технічний університет України «Київський політехнічний інститут»
Інститут телекомунікаційних систем
пров. Індустріальний, 2, Київ, 03056, Україна

poooshta@ukr.net; alexeyev@its.kpi.ua

Анотація. *Проведено порівняльний аналіз сучасних технологій, які направленні на вирішення задач обробки структурованих і неструктурованих даних значних обсягів, так названих «Великих Даних». Зроблені висновки про доцільність вибору конкретної технології у залежності від типів задач аналізу даних та їх особливостей.*

Ключові слова

Обробка даних великого обсягу, паралельного обчислення, порівняльний аналіз, MapReduce

1 Вступ

Проблема збору даних, їх інтеграції та аналізу гостро стоїть на даний час. Саме тому питання використання технологій масивно-паралельних СКБД, рішень класу бізнес-аналізу (Business Intelligence), нетрадиційних СКБД NoSQL та альтернативних рішень для побудови систем, що забезпечують розподілену обробку даних, є дуже актуальним. Серед таких рішень особливою популярністю користується модель MapReduce, і побудований на її концепціях програмний каркас Hadoop. Згідно концепції технології обробка даних розділяється на велику кількість елементарних завдань, які виконуються на різних вузлах кластера і, в остаточному підсумку, зводиться в єдиний результат. В статті здійснено порівняльний аналіз MapReduce технології з традиційними рішеннями в обробці даних великого об'єму.

2 Порівняння MapReduce з іншими системами

Програмна конструкція MapReduce, розроблена Google, відома як спрощена обробка даних на великих кластерах. MapReduce - це ширше поняття, аніж програмний каркас (framework), так як являє собою інфраструктурне рішення, здатне ефективно використовувати сьогодні кластерні, а в майбутньому багатоядерні архітектури, своєю перспективністю вона і привертає до себе підвищену увагу. В той же час виникає необхідність в обґрунтування вибору технології MapReduce серед інших рішень для здійснення обробки великих масивів даних. В порівнянні з традиційним підходом для обчислень з використанням СКБД та HPC основними критеріями є розмірність даних (робота з Великими Даними), масштабованість системи, тип обчислень та даних, можливість виконання задач автоматично з метою зменшення професійних вимог до спеціаліста.

2.1 Системи керування базами даних

Традиційним рішенням для використання в аналітичних системах і сховищах даних з обсягами даних від сотень гігабайт до сотень терабайт були реляційні СКБД. На противагу рішенню в аналізі даних СКБД з великою кількістю дисків – накопичувачів, MapReduce виграє на апаратному рівні. Причина в особливості роботи накопичувачів на жорстких дисках, а саме те, що час пошуку даних прискорюється повільніше, аніж швидкість їх передачі. Процес пошуку характеризується затримкою дискових операцій, в той час як швидкість передачі даних залежить від пропускної здатності диска. Якщо модель доступу до даних переважає над пошуком, то процес займе більше часу, щоб зчитати або записати великі частини набору даних, аніж час, затрачений на прохід через їх, який управляється швидкістю передачі даних.

У багатьох відносинах MapReduce можна розглядати як доповнення до СКБД . MapReduce добре підходить для задач, в яких необхідно проаналізувати весь набір даних в пакетному представленні. Перевагу

СКБД слід надати при роботі з одиночними запитами або оновленнями, в яких набір даних був проіндексований для доставки з низькою затримкою на пошук і часом оновлення для відносно невеликих об'ємів даних . З іншого боку, для оновлення невеликої частини записів в базі даних ефективно використовуються традиційні B-Tree (структури даних в реляційних базах даних, які обмежені по швидкості виконання пошуку). Проте при роботі з більшості баз даних, B-Tree виявились менш ефективними, ніж MapReduce, який використовує сортування – злиття (Sort-Merge) для перебудови бази даних. Тобто, СКБД MapReduce підходить для додатків, де дані записуються один раз, а зчитуються багато разів, в той час як реляційні бази даних, ефективні для обробки наборів даних, які постійно оновлюються.

Реляційні дані часто нормалізуються для забезпечення своєї цілісності і видалення надлишковості. В MapReduce зчитування записів є нелокальною операцією, що робить використання нормалізації проблематичним.

Log веб-сервера є вдалим прикладом великої кількості ненормованих записів - це обґрунтовує зручність аналізу log - файлів будь-якого типу за допомогою MapReduce. MapReduce є моделлю з лінійною масштабованістю [1]. Важливо, що для написаних програмістом функцій map та reduce не важливі особливості кластера, на якому здійснюється обробка чи розмір даних, які опрацьовуються. Вони можуть залишатися незмінними як для невеликого набору даних чи для масиву. Особливість в тому, що рішення подвоїти розмір вхідних даних призведе до уповільнення виконання завдання вдвічі (рис.1 а). В той же час подвоєння розміру кластера поверне попередню швидкість, що не властиво для SQL запитів (рис.1 б). Дані графіки отримані на основі представлених результатів SRA International, Inc. в ході дослідження ефективності технології MapReduce[2]

Однак, відмінності між реляційними базами даних і MapReduce системи стираються - як і в реляційних базах даних почали враховувати ідеї з MapReduce (наприклад, бази даних Greenplum, засновані на доопрацьованій PostgreSQL для бази даних з масивно-паралельною архітектурою), так і, з іншого боку, на основі MapReduce розробили мови запитів високого рівня (такі, як Pig і Hive).

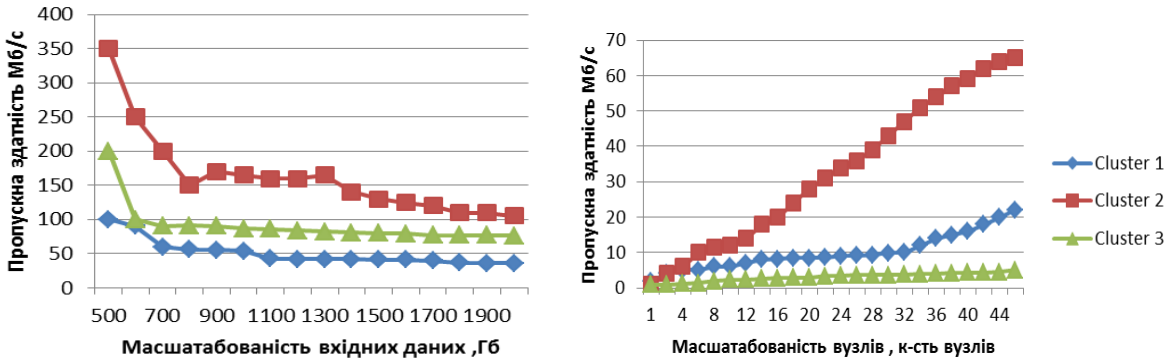

Рис.1 Графіки залежності пропускної здатності для кластерів різної конфігурації (Cluster 1: 2 ГГц Intel Xeon Processor L5508/4GB RAM/2x160GB HD;Cluster 2: 2 ГГц Intel Xeon Processor L5508/4GB RAM/2x160GB HD;Cluster 3: 3,2 ГГц Intel Xeon Processor W5580/16 GB RAM/2x320GB HD) від масштабованості (а) вхідних даних (б) кількості вузлів в кластері

2.2 Комп'ютерні обчислення, реалізовані на ГРІД-системах

Високопродуктивні обчислення (HPC) і співтовариство Grid Computing здійснювали великомасштабні обробки даних протягом багатьох років, використовуючи такі API, як Message Passing Interface (MPI). Такий підхід ефективно працює переважно з інтенсивними обчисленнями, але виникають проблеми, коли вузли повинні отримати доступ до великих обсягів даних (для розмірів від сотні гігабайт), так як пропускна здатність мережі є слабким місцем і обчислювання на вузлах переходить в стан простою. MapReduce розподіляє дані так, щоб вони знаходились поряд з вузлом, який здійснює обчислення, з метою забезпечити швидкий доступ, оскільки доступ до даних є локальним. Всі MapReduce-реалізації строго притримуються цієї властивості явно моделюючи топологію мережі, так як пропускна здатність мережі є найціннішим ресурсом в середовищі дата – центру.

Використання MPI дозволяє програмісту контролювати процес, але натомість вимагає від нього явної обробки механіки потоку даних доступних через сокети та низькорівневі підпрограми C. MapReduce працює тільки на високому рівні [3]: програміст обдумає функції пари *<ключ,значення>*, але при цьому потік даних не є явним. Проблематичною являється координація процесів для великомасштабних розподілених обчислень, особливо, коли мова йде про обробку часткової відмови. За такої відмови продовжує виконуватися загальне обчислення. За рахунок того, що MapReduce має нерозподілену архітектуру, програміст обробляє невиконаний

процес та перенаправляє виконання на робочі машини тоді, коли не виконано map чи reduce завдання . Таким чином, з точки зору програміста, порядок, в якому повинна явно контролювати своєю реєстрацією наведення і відновлення. В такому випадку, основний контроль реалізує програміст, що ускладнює написання MPI програм.

В той же час MapReduce програє MPI програмам, коли мова йде про ітеративні обчислення. Приведені графіки отримані на основі представлених даних в ході наукового експерименту департаменту комп'ютерних наук Indiana University. Для аналізу продуктивності MapReduce та MPI програм в контексті цього експерименту розглядаються наступні задачі: кластеризація за допомогою K-means алгоритму та перемноження матриць. Перемноження матриць ілюструє неефективність здійснення ітеративних обчислення MapReduce програмами (рис.2 а). Пояснюється даний факт відсутністю можливості зберігати статичні дані в пам'яті між викликами. Для кожної ітерації при перемноженні матриць A та B, всі map-завдання отримують на виході: (I) колонки блоку матриці B, і (II), блок рядків матриці; на виході маємо рядок результуючої матриці C. Блок стовпців пов'язаний з конкретним map-завданням, яке фіксується протягом обчислень, в той час як блок рядків змінюються в кожній ітерації. В моделі програмування в Hadoop (типова модель MapReduce), немає можливості зберігати статичні дані між викликами. Таким чином, він завантажує обидва блоки стовпчика і рядка блоку в кожній ітерації обчислень, що знижує продуктивність в порівнянні с MPI- програмами.

K-means кластеризація є наглядним прикладом задач з декількома ітеративними обчисленнями, які виконуються одночасно для загального обчислення. З цієї цілю обрано алгоритм кластеризації колекції 2D точок даних. Для виконання 16 ітерацій для 5.12E+06 точок виконання MPI програми проходить на два порядки швидше (рис.2 б)

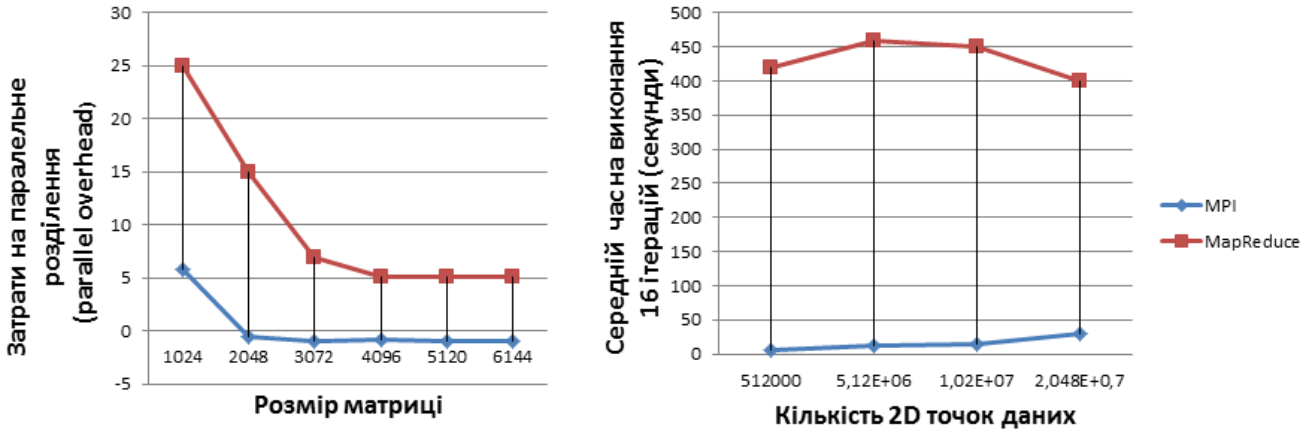

Рис.2 (а) Затрати на паралельне розподілення програми MapReduce та MPI для перемноження матриць (б) Продуктивність у MapReduce та MPI для K-means кластерних обчислень

2.3 Волонтерські обчислення [4]

В певних питаннях схожі до MapReduce завдання вирішують проекти групи @Home Network. В них добровольці надають у безкоштовне використання системою розподілених обчислень процесорний часу власних комп'ютерів. Ідея розбиття проблеми на самостійні частини, які окремо опрацьовуються, присутня і в MapReduce, проте існує суттєва відмінність: ресурсомісткість (CPU-intensive) проектів @Home Network робить його придатним лише для роботи на сотнях тисяч комп'ютерах по всьому світу, за умови коли час передачі одиниці роботи незначний в порівнянні з часом її виконання. MapReduce призначений для виконання завдань, які тривають кілька хвилин або годин на спеціалізованих апаратних засобах в єдиному центрі обробки даних з дуже високою сукупною пропускною здатністю з'єднань.

3 Висновки

Згідно з визначеними критеріями був здійснений порівняльний аналіз технологій обчислення великих масивів даних результати якого підсумовуються в табл.1

Представлення проблеми в формі Map-Reduce дозволяє відносно легко розпаралелювати обчислення, направляти дані до процесорів і балансувати навантаження між ними. Деталі всіх цих питань можуть бути приховані від користувача, а можливості паралелізму на рівні задач та рівні команди легко ідентифікуються.

MapReduce створена з розрахунку на використання кластерної апаратної архітектури для вирішення задачі паралельної роботи з Великими Даними. Застосування даного підходу обробки даних на противагу традиційним рішенням обґрунтовано такими перевагами як висока продуктивність та можливість опису

обробки зрозумілим кодом. Як показав проведений аналіз, можливості коду MapReduce набагато ширші за SQL, навіть без використання спеціалізованих рішень.

Представлений огляд даних технологій розподілених обчислень обґрунтовує вибір саме технології MapReduce для здійснення обробки великих масивів даних у формі не ітеративних алгоритмів, де вузли потребують невеликого обміну інформацією (не ітеративні та не залежні). Масивно-паралельні СКБД виграють у роботі з структурованими даними, які часто записуються, а також за рахунок інтегрованості системи. Надати перевагу MPI системам варто при необхідності контролю процесу в тонкій деталізації та інтенсивних обчисленнях задач, в той час, як MapReduce добре зарекомендував себе в обробці задач з великим об'ємом даних.

У подальшій роботі планується дослідити реалізації моделі MapReduce на C#, Ruby, Java на прикладі каркасу Hadoop для створення програмних додатків розподілених інформаційних систем. Очікується, що конвергенція MapReduce з іншими технологіями для обробки великих масивів даних дозволить підвищити продуктивність створених програмних додатків.

Табл 1. Порівняльна характеристика використання MPI, СКБД та MapReduce

Властивості	**Технології**		
	MapReduce	**GridComputing**	**Традиційні СКБД**
Середній розмір даних	Петабайти	Петабайти	Гігабайти
Пакетний доступ	Так	Так	Так
Масштабованість	Лінійна	Лінійна	Нелінійна
Балансування навантаження	Автоматично за допомогою бібліотеки MapReduce	Розробка паралельного розподілення реалізується програмно	Так
Локальна оптимізація	Так	Ні	Частково
Оновлення даних	Записати один раз, зчитувати багато раз	Записувати та зчитувати багато раз	Переваги в роботі з одиничними запитами, які постійно оновлюються.
Паралельне розподілення ресурсів	Автоматично	Програмно	В залежності від архітектури share-everything, share-disk и share-nothing, як програмно, так і паралельно
Тип обчислення	Неітеративні алгоритми	Інтенсивні обчислення	-//-
Робота з неструктурованими даними	Так (не вимагає використання схеми)	Так	Ні (Вимагає нормалізації)
Структура	Динамічна схема	Будь-яка	Статична схема
Вимоги до спеціаліста	Програміст без досвіду роботи з паралельними та розподіленими системами	Вимагає від спеціаліста явної обробки механіки потоку даних доступних через сокети та низькорівневі підпрограми C	Аналогічні до Map операції , попри те що деякі СКБД підтримують визначенні користувачем функції (UDFs), складно представити в виді SQL

Література

[1] Tom White *«Hadoop: The Definitive Guide»* O'Reilly Media.-Boston,USA,2009, с.72-92

[2] Guest post from Paul Burkhardt, a Research Developer at SRA International, Inc.
A profile of Apache Hadoop MapReduce computing efficiency (part 2) by Jon Zuanich
resource: http://blog.cloudera.com/blog/2010/12/a-profile-of-hadoop-mapreduce-computing-efficiency-continued/

[3] Jackson H.C. Yeung1, C.C. Tsang1, K.H. Tsoi1, Bill S.H. Kwan1, Chris C.C. Cheung2, Anthony P.C. Chan2 and Philip H.W. Leong *«Map-reduce as a Programming Model for Custom Computing Machines.»* in Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), 2002, pp. 13–21.

[4] Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Leboisky, M., "SETI@home-Massively Distributed Computing for SETI," Computing in Science and Engineering 3, 78–83 (Jan/Feb 2001).