Перспективы и проблемы использования оперативной энергонезависимой памяти

С.Д. Кузнецов <kuzloc@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, дом 25
Московский государственный университет имени М.В. Ломоносова,
119991 ГСП-1 Москва, Ленинские горы,
МГУ имени М.В. Ломоносова, 2-й учебный корпус, факультет ВМК
Московский физико-технический институт,
141700, Московская область, г. Долгопрудный, Институтский пер., 9
Высшая школа экономики,
101000, Москва, ул. Мясницкая, д. 20

Аннотация. В начале статьи демонстрируется, что технология наиболее распространенных в настоящее время SQL-ориентированных СУБД неразрывно связана с технологией HDD. Особенности HDD влияют на структуры данных и алгоритмы выполнения операций, методы управления буферным пулом СУБД, управление транзакциями, оптимизацию запросов и т.д. Альтернативой дисковым СУБД являются in-memory СУБД, хранящие базы данных целиком в основной памяти. Несмотря на наличие у in-memory СУБД ряда преимуществ перед дисковыми СУБД, в настоящее время конкуренция между практически отсутствует. Это, прежде всего, связано с естественными ограничениями на размеры баз данных, свойственными in-memory СУБД. В настоящее время появились новые виды аппаратуры хранения данными: SSD – блочные твердотельные накопители и SCM – энергонезависимая основная память. Характеристики SSD делали целесообразной разработку СУБД в расчете на их исключительное использование, однако до сих пор такая СУБД не создана, а SSD используются просто вместо HDD в СУБД, не учитывающих их особенности. Наличие SCM позволяет радикально упростить архитектуры СУБД и значительно повысить их производительность. Для этого нужно пересмотреть многие идеи, используемые в дисковых СУБД.

Ключевые слова: SQL-ориентированные СУБД, магнитные диски с подвижными головками, оценочная оптимизация запросов, СУБД с хранением баз данных в основной памяти, твердотельные накопители на флэш-памяти, энергонезависимая основная память

1. Введение

Технология наиболее распространенных SQL-ориентированных («реляционных») систем управления базами данных (СУБД) неразрывно связана с технологией устройств хранения данных на магнитных дисках с подвижными головками (Hard Disk Drive, HDD). Первые HHD были выпущены компанией IBM в 1956 г. В технологии HDD преодолевались недостатки ранних устройств хранения данных – магнитных лент (magnetic tape data storage, чисто последовательный доступ) и магнитных барабанов (drum memory, ограниченная емкость), обеспечивая меньшую, чем у магнитных лент, но значительно большую, чем у магнитных барабанов, емкость, а также меньшую, чем у магнитных барабанов, но значительно большую, чем у магнитных лент, скорость выполнения произвольных обменов данными между основной и внешней памятью. Если добавить к этому умеренную стоимость HDD, то эти устройства являлись вполне подходящими для хранения баз данных.

На технологию СУБД повлияли технологические особенности HDD. Во-первых, HDD обеспечивают внешнюю память, обмены с которой обычно производятся блоками байт одного и того же размера. Эта особенность приводит, как минимум, к двум архитектурным решениям. (1) Для хранения баз данных и убыстрения обработки запросов выбираются

структуры данных и алгоритмы выполнения операций, для которых естественна блочная природа внешней памяти. В частности, для организации индексов наиболее часто применяются разновидности B-деревьев [1]. (2) Для обеспечения доступа к данным программам СУБД и сглаживания относительно небольшой скорости выполнения произвольных обменов с внешней памятью и относительно высокой скорости обработки данных в основной памяти СУБД производит собственную буферизацию (кэширование) блоков внешней памяти базы данных в основной памяти [2, subsection 3.3, Buffer Management, 3, п. 10.1.1 Управление буферным пулом базы данных.].

Во-вторых, при выполнении обменов с внешней памятью HDD дисковая аппаратура выполняет три основных операции: подвод головок к требуемому цилиндру дискового пакета (seek), прокручивание дискового пакета на требуемое угловое расстояние (latency), чтение или запись данных с их передачей в основную память или из нее (data transfer). При выполнении произвольного обмена время выполнения первых двух операций исчисляется миллисекундами, а это означает, что время чтения произвольного блока данных из внешней памяти или его записи на несколько десятичных порядков больше времени выполнения соответствующего цикла переписи в основной памяти. Поэтому при выполнении любой операции уровня SQL над базой данных определяющим накладным расходом является число требуемых обменов с внешней памятью. На этом наблюдении основана оценочная оптимизация запросов (cost-based query optimization), применяемая во всех развитых SQL-ориентированных СУБД и основанная на пионерской работе [4].

Приведенных замечаний достаточно, чтобы убедиться в глубокой зависимости наиболее распространенной технологии SQL-ориентированных СУБД от особенностей HDD. Ориентация на использование этих устройств хранения данных влияет как на общую архитектуру СУБД, так и на выбор основных структур данных и алгоритмов.

В конце 1970-х – 1980-х гг. предпринимались попытки создания специализированной аппаратуры для поддержки СУБД, включая аппаратуру хранения данных на дисках с фиксированными головками (head-per-track disk). Более того, имелись прототипы таких устройств, в которых в магнитные головки встраивались специальные микропроцессоры, фильтрующие данные «на лету» при их считывании с диска (processor-per-track systems и processor-per-head systems) [5]. Однако к началу 1990-х стала ясна бесперспективность такого подхода [6], и на протяжении следующих двух десятилетий технология СУБД базировалась главным образом на устройствах хранения данных категории HDD.

В то же время появилась и развилась альтернативная технология СУБД с хранением баз данных в обычной энергозависимой основной памяти (in-memory DBMS) [7]. В таких СУБД структуры данных и алгоритмы выполнения операций отличаются от используемых в дисковых СУБД. В частности, при выборе структур данных нужно учитывать наличие кэш-памяти в процессорах [8]. Должны отличаться и принципы оптимизации запросов, хотя информация об оптимизаторах запросов в in-memory СУБД в доступной литературе отсутствует.

Вероятно, наиболее зрелыми представителем этой категории СУБД является TimesTen [9], существующая с 1996 г. и приобретенная Oracle в 2005 г., и solidDB [10], существующая с 1992 г. и приобретенная IBM в 2007 г. Эти системы поддерживают очень быстрое выполнение запросов к базам данных (поскольку база данных и все индексы целиком сохраняются в основной памяти), однако для выполнения операций изменения баз данных требуются обращения к внешней памяти, так что скорость выполнения таких операций не отличается от соответствующей скорости СУБД, хранящих базы данных на дисках.

Особняком стоит in-memory СУБД VoltDB [11], являющаяся транзакционной массивно-параллельной системой без общих ресурсов между узлами (shared nothing). В этой системе свойство долговечности (durability) транзакций поддерживается на основе репликации данных в нескольких узлах, а внешняя память вообще не используется. Подробности организации VoltDB (и ее прототипа H-Store) см. в [12].

Следует заметить, что несмотря на наличие у in-memory СУБД ряда преимуществ перед дисковыми СУБД, в настоящее время конкуренция между практически отсутствует. Это, прежде всего, связано с естественными ограничениями на размеры баз данных, свойственными in-memory СУБД.

В первые десятилетия 21-го века в технологии аппаратных средств хранения данных произошли (и продолжают происходить) существенные изменения. Появились так называемые блочные твердотельные накопители, основанные на технологии флэш-памяти (Solid-State Drive, SSD), и сравнительно быстро догнавшие HDD по показателю максимальной емкости (до 16 терабайт в 2016 г.), превосходя их по ряду других показателей (проигрывая в основном только в цене). В следующем разделе будут кратко обсуждены потенциальные возможности применения SSD в архитектуре СУБД, компоненты СУБД, на которые должен был бы максимально подействовать переход от HDD к SSD, а также реальное состояние дел в технологии СУБД через 10 лет после того, как SSD на флэш-памяти стали реально доступны.

В последние годы полностью реальной стала перспектива появления на рынке оперативной энергонезависимой памяти (Non Volatile Random Access Memory, NVRAM), которую, возможно, более выразительно, хотя и слишком длинно по-русски называют основной памятью с возможностью долговременного хранения данных (Storage Class Memory, SCM). Такая память допускает байтовую адресацию, прямо доступна для команд процессоров, но при этом сохраняет содержимое после отключения электропитания.

Использование SCM открывает путь к построению СУБД, основанных на одноуровневой памяти. Эти СУБД могут оказаться гораздо быстрее дисковых при более простой организации. Перспективам появления таких СУБД и имеющимся проблемам посвящен третий раздел статьи.

Четвертый раздел завершает статью и содержит заключительные замечания.

2. SSD на флэш-памяти и технология СУБД

Как и HDD, SSD – это блочное внешнее запоминающее устройство, сохраняющее данные после выключения электропитания. Основными отличиями SSD от HDD являются следующие:

- в SSD отсутствуют механические компоненты, поэтому для любого блока скорость выполнения обмена с SSD одна и та же;
- если среднее время обмена с произвольным блоком HDD составляет около 10 миллисекунд как для чтения, так и для записи, то время чтения произвольного блока в современных SSD – около 20 микросекунд (на три десятичных порядка меньше, чем у HDD), а время записи – около 200 микросекунд (на два десятичных порядка меньше, чем у HDD);
- пока SSD стоят дороже, чем HDD (на 2016 г. примерно в 10 раз), но стоимость HDD в пересчете на терабайт объема поддерживаемой памяти в последние годы стабилизировалась, а SSD дешевеют;

- в настоящее время SSD являются существенно менее надежными устройствами, чем HDD.

2.1 SSD-ориентированные СУБД

Только последняя в списке характеристика может в принципе препятствовать полномасштабному применению SSD в СУБД. Непонятно, удастся ли разработчикам аппаратуры SSD избавиться от этого недостатка, но первые две характеристики кажутся настолько привлекательными, что еще 10 лет тому назад я пытался (не слишком успешно) убедить своих студентов заняться исследованиями архитектуры СУБД, в которой для хранения баз данных используются SSD.

Понятно, что в наибольшей степени особенности SSD могли бы повлиять на управление внешней памятью, управление буферами основной памяти и оптимизатор запросов. В существующих дисковых СУБД, поскольку при выполнении запросов часто приходится производить полный просмотр таблиц без использования индексов, стремятся располагать на диске блоки одной таблицы так, чтобы при переходе от текущего блока к следующему не требовалось сильно перемечать магнитные головки. В СУБД, основанной на использовании только SSD, блоки одной таблицы могут располагаться во внешней памяти произвольным образом.

Время записи блока во внешнюю память SSD на десятичный порядок больше времени чтения блока за счет потребности предварительной подготовки сектора внешней памяти, в который будет производиться запись [13]. При управлении буферами основной памяти в СУБД, ориентированной на использование SSD, имеет смысл заранее подготавливать к записи сектора внешней памяти и при выталкивании из буфера во внешнюю память измененного образа ранее прочитанного блока внешней памяти писать его не в тот сектор, из которого он был прочитан, а в некоторый сектор, уже подготовленный к записи.

Но распределение внешней памяти и управление буферами основной памяти – это мелочи по сравнению с оптимизацией запросов. Как отмечалось во введении, современные оценочные оптимизаторы основываются на предположении, что произвольные обмены с внешней памятью выполняются так долго, что стоимость плана выполнения запроса можно оценивать числом требуемых для этого обменов, пренебрегая временем, которое потребуется для процессорной обработки данных. Чтение из внешней памяти SSD выполняется в 1000 раз быстрее, чем с использованием HDD. Поэтому при переходе от HDD к SSD это предположение нужно было бы подвергнуть строгой ревизии.

Имеется в виду, что прямой перенос оценок планов выполнения запросов из среды HDD в среду SSD может привести к плачевным результатам. Неправильный учет временных затрат на обмены с внешней памятью и процессорную обработку данных в основной памяти может привести к выбору оптимизатором запросов заведомо не оптимальных планов выполнения запросов, что приведет к недоиспользованию потенциала SSD. Конечно, запросы не станут выполняться медленнее, чем при применении HDD, но ради этого не стоит менять аппаратуру управления внешней памятью. Другими словами, для эффективного использования SSD оптимизаторы запросов нужно значительно переделывать.

Несмотря на привлекательность идеи замены HDD на SSD в аппаратной поддержке СУБД, практически отсутствуют проекты (как коммерческие, так и исследовательские) по разработке SSD-ориентированных СУБД. Мне удалось обнаружить только проект

FlashyDB, выполняемый в немецком университете Ройтлингена [14]. Объявлены следующие цели проекта:

- исследовать влияние SSD на основе флэш-памяти на архитектуры и производительность существующих систем баз данных, реляционных хранилищ данных (data warehouse) и систем с поколоночным хранением таблиц (column store);
- разработать алгоритмы и структуры данных, обеспечивающие оптимальное использование характеристик SSD на основе флэш-памяти в сценариях OLTP и OLAP;
- реализовать прототип системы.

Список исследовательских тем, затрагиваемых в проекте, включает архитектуры систем баз данных, обработку транзакций, управление мультидоступом, восстановление после сбоев, управление буферами, индексация, оптимизация запросов, размещение данных. Как видно, направленность проекта вполне соответствует высказанным выше соображениям. По-видимому, одной из первых статей, посвященных проекту FlashyDB, была статья [15]. Полный список опубликованных статей доступен на сайте проекта [14]. Как показывает этот список, далеко не во всех намеченных направлениях исследований получены существенные результаты.

Возможно, недостаточная активность исследователей по построению истинных SSD-ориентированных СУБД связана с тем, что до недавнего прошлого максимальная емкость устройств хранения данных во флэш-памяти ограничивалась одним терабайтом. Однако технология быстро развивается, и уже в 2016 г. компания Samsung представила SSD емкостью 32 Тб и обещает довести емкость своих SSD до 100 Тб. Seagate показала SSD емкостью 60 Тб. Думаю, это «подстегнет» сообщество баз данных.

2.2 Двухуровневый кэш на основе SSD

Пока же емкость SSD была сравнительно невелика, достаточно популярной была идея использования SSD в составе иерархического двухуровневого буфера в традиционных СУБД, ориентированных на использование HDD [16]. Суть идеи достаточна проста. Если мы по каким-то причинам хотим продолжать использовать в СУБД для хранения баз данных HDD, но при этом получать достаточную пользу от применения SSD, то почему бы временно не хранить во флэш-памяти часть блоков базы данных, которая, вероятно, требуется в данный момент времени.

Для реализации этой идеи достаточно изменить лишь один компонент традиционной дисковой СУБД – менеджер буферов в основной памяти. Буфер становится двухуровневым: кэш первого уровня размещается в основной памяти, а кэш второго уровня – во флэш-памяти SSD. Блоки базы данных, требуемые для выполнения операций над базой данных, считываются из дисковой внешней памяти в буферные страницы кэша первого уровня. При нехватке памяти в кэше первого уровня происходит замещение какой-либо буферной страницы. Если ее содержимое изменялось после чтения из внешней памяти, то страница перемещается в кэш второго уровня (с учетом замечаний об управлении буферами из подраздела 2.1). Если не хватает памяти в кэше второго уровня, то замещаемый блок перемещается во внешнюю память HDD.

В [16] приводится обзор алгоритмов управления подобным двухуровневым буферным пулом. Все разработанные алгоритмы являются сложными и ресурсоемкими. Мне неизвестна какая-либо СУБД, в которой эти алгоритмы реально бы применялись. Тем не менее, по-видимому, внедрение в состав дисковой СУБД двухуровневого кэша с

использованием SSD – это наиболее дешевый способ модификации СУБД с целью повышения ее производительности за счет применения технологии SSD.

В этом случае в кэш второго уровня постепенно попадают наиболее часто используемые блоки базы данных, доступ к которым затем происходит со скоростью, свойственной SSD. Кроме того, поскольку флэш-память является энергонезависимой, не требуются выталкивания из памяти SSD в память HHD ни в каких случаях, кроме нехватки места.

Однако этот подход не отменяет потребность в разработке чистых SSD-ориентированных СУБД, в которых особенности характеристик системы хранения данных учитываются во всех компонентах.

2.3 Гибридные диски

Самый простой способ получить какой-то выигрыш в производительности СУБД от применения технологии SSD состоит в том, чтобы просто заменить аппаратуру HDD на аппаратуру SSD без каких-либо изменений СУБД. Как отмечалось в подразделе 2.1, операции над базами данных после этого гарантированно не будут выполняться медленнее, а скорее всего, будут выполняться в среднем быстрее.

При наличии баз данных большого объема смена аппаратных средств хранения данных обойдется явно недешево, и смутные обещания лучшей жизни (на качественном уровне) вряд ли могут сподвигнуть менеджеров компаний на такие расходы. В гибридных устройствах хранения данных на жестких дисках (solid-state hybrid drive, SSHD) совместно используются технологии SSD и HDD.

В SSHD SSD используется для кэширования содержимого блоков HDD, к которым наиболее часто происходят обращения. В результате SSHD часто работает со скоростью SSD при стоимости, близкой к стоимости HDD. Попробовать повысить производительность СУБД за счет перехода от использования HDD к использованию SSHD стоит уже не так дорого, хотя, конечно, это решение не опирается на какие-либо технологические доводы и остается рискованным.

3. Оперативная энергонезависимая память: перспективы для СУБД

В настоящее время реальные решения SCM могут обеспечить три технологии: память на основе фазового перехода (Phase-Change Memory, PCRAM) [17], резистивная память с произвольным доступом (Resistive Random-Access Memory, RRAM) [18] и магниторезистивная оперативная память (Magnetoresistive Random-Access Memory, MRAM) [19].

PCRAM основывается на поведении халькогенида[1], который при нагреве может «переключаться» между двумя состояниями: кристаллическим и аморфным. Кристаллическое и аморфное состояния халькогенида кардинально различаются электрическим сопротивлением. Аморфное состояние, обладающее высоким сопротивлением, используется для представления двоичного 0, а кристаллическое состояние, обладающее низким уровнем сопротивления, представляет 1.

Основная идея RRAM состоит в том, что диэлектрики, которые в нормальном состоянии имеют очень высокое сопротивление, после приложения достаточно высокого напряжения могут сформировать внутри себя проводящие нити низкого сопротивления, и по сути

[1] Бинарные химические соединения халькогенов (элементов 16-й группы периодической системы, к которым относятся кислород, сера, селен, теллур, полоний и ливерморий) с металлами

превратиться из диэлектрика в проводник. Эти проводящие нити могут образовываться с помощью разных механизмов. С помощью приложения соответствующих уровней напряжения проводящие нити могут быть как разрушены (и материал снова станет диэлектриком), так и сформированы снова (и материал опять станет проводником).

Данные в MRAM хранятся в магнитных элементах памяти. Магнитные элементы сформированы из двух ферромагнитных слоёв, разделенных тонким слоем диэлектрика. Один из слоёв представляет собой постоянный магнит, намагниченный в определённом направлении, а намагниченность другого слоя изменяется под действием внешнего поля. Устройство памяти организовано по принципу сетки, состоящей из отдельных «ячеек», содержащих элемент памяти и транзистор.

Не буду останавливаться на том, какие компьютерные компании предпочитают ту или иную технологию SCM. Уже пару лет разные крупные компании обещают в ближайшем начать производство соответствующих чипов. Пока на рынке появились SSD, основанные не на флэш-памяти, а на SCM с блочными обменами. Думаю, что соответствующая оперативная память появится не позже следующего года.

Интересно, что еще в 2011-м г. государственная корпорация Роснано заключила с французской компанией Crocus соглашение о налаживании в России «производства памяти MRAM средней и высокой плотности с проектными нормами 90 и 65 нм» [19]. Для справедливости следует отметить, что Samsung планирует начать массовое производство такой памяти на основе 28-нанометровой технологии [20].

Тем не менее, выбор для производства в России именно памяти MRAM, по-видимому, оправдан, поскольку у MRAM ожидается время чтения и записи около 20 нс (меньше, чем у сегодняшней DRAM) при долговечности, соизмеримой с долговечностью DRAM и HDD, а у PCRAM и RRAM время чтения в несколько раз больше (а запись медленнее чтения), а долговечность значительно меньше [21].

Конечно, до появления разных видов SCM на рынке невозможно достоверно сравнивать их характеристики, но имеется надежда, что MRAM с обещанными характеристиками действительно появится и далее в статье я на это полагаюсь.

Следует также обратить внимание, что энергонезависимая оперативная память будет использоваться в компьютерах, процессоры которых оснащены вполне энергозависимыми кэшами. Чтобы обеспечить возможность фиксации транзакций в SCM, в систему команд процессоров Intel были добавлены две команды – CLWB и CLFLUSH. Обе команды предназначены для выталкивания данных из стеков всех уровней в SCM, но первая команда сохраняет выталкиваемые данные в стеке, а вторая вынуждает при следующем обращении считывать данные из SCM.

3.1 SQL-ориентированные СУБД на основе SCM

На первый взгляд, в качестве основы для разработки СУБД, в которой для хранения данных используется только SCM (и вовсе не используется внешняя память) было бы разумно использовать какую-либо имеющуюся in-memory СУБД. Действительно, in-memory СУБД, как и СУБД на основе SCM, сохраняют базы данных целиком в основной памяти. В расчете на это выбираются основные структуры данных и алгоритмы выполнения операций, в расчете на это строится оптимизатор запросов (или, вернее, должен был бы строиться, поскольку достоверной информации об оптимизаторах запросов в существующих in-memory СУБД мне получить не удалось).

Однако между in-memory СУБД и СУБД на основе SCM имеется принципиальное различие, которое не позволяет так просто использовать имеющиеся решения: in-memory СУБД рассчитаны на использование традиционной энергозависимой основной памяти, а СУБД на основе SCM – на использование энергонезависимой основной памяти. Для поддержки свойства долговечности (durability) транзакций в in-memory СУБД используется внешняя память (HDD или SSD – здесь не принципиально), т.е. как и дисковых СУБД используется двухуровневая иерархия памяти, на первом уровне которой находится энергозависимая основная память, а на втором – энергонезависимая внешняя память. В отличие от дисковых СУБД, в этом случае основная память хранит всю базу данных (а не служит кэшем), а внешняя память служит для поддержки долговечности хранения баз данных.[2]

При разработке СУБД на основе SCM мы имеем дело с принципиально одноуровневой средой хранения баз данных, обладающей возможностью байтовой адресации. В этом случае мы, вообще говоря, можем полностью отказаться от блочной структуры памяти и начать распределять ее (для всех целей, связанных с поддержкой баз данных) порциями произвольного размера. Стоит задуматься над тем, может ли это оказаться полезным и если да, поразмышлять о распределении основной энергонезависимой памяти фрагментами произвольного размера: как бороться в внешней фрагментацией? допустимы ли сдвижки памяти? не стоит ли воспользоваться какой-либо разновидностью метода близнецов (например, методом фибоначчиевых близнецов [23, 12.5 – Методы близнецов])? и т.д.

Если отсутствует блочная структура памяти, нет причин использовать для организации индексов B-деревья.[3] Что использовать вместо B-деревьев? Стоит ли попытаться воспользоваться каким-либо методом поиска в основной памяти на основе деревьев [24, 4.1 – Методы поиска в основной памяти на основе деревьев]? Может быть, лучше применить какой-либо метод поиска на основе хэширования [24, 4.2 – Методы хэширования для поиска в основной памяти]? Или же лучше поискать или придумать что-нибудь новенькое?

Как поддерживать сериализацию транзакций в транзакционных системах? Использовать ли версионные алгоритмы и какими они должны быть в данном случае? Стоит ли в СУБД на основе SCM экономить на сборке мусора, потребность в котором возникает, если не ограничивать число версий объектов баз данных? Как вести журнал в SCM? Нужны ли логический и физический журналы? Какова единица записи в физический журнал?

Наконец, как оптимизировать запросы? Как строить оценочные функции?

Вопросов великое множество, и на все их нужно уметь правильно отвечать, чтобы получить реальные преимущества от разработки СУБД на основе SCM. К сожалению, хотя потребность в энергонезависимой основной памяти отмечал еще в 1987 г. Майкл Стоунбрейкер при разработке Postgres [25], в настоящее время проекты по полномасштабной разработке СУБД на основе SCM практически отсутствуют. Это, в частности, подтверждается тем, что на конференции SIGMOD в 2017 г. тьюториал «Как построить систему управления базами данных в основной энергонезависимой памяти» [21] представляли Джой Арулрадж и Эндрю Павло из Карнего-Меллонского университета, являющиеся лидерами проекта Peloton [26].

---

[2] Как отмечалось в разд. 1, особый случай представляет СУБД VoltDB, но она принципиально работает в режиме shared nothing в массивно-параллельной среде.

[3] Вообще, кажется странным использование B-деревьев в in-memory СУБД – все-таки по своей природе это дисковая структура памяти.

В списке основных характеристик проекта числится изначальная поддержка технологии хранения данных на основе основной энергонезависимой памяти. К сожалению (с позиций человека, стремящегося развитию этой технологии), как показывает название проекта, эта цель проекта не является основной. Основной целью является интеграция компонентов искусственного интеллекта для обеспечения возможности автономных (само)оптимизаций системы в зависимости от текущей рабочей нагрузки [27]. Эта задача также очень актуальна, но если учесть, что в настоящее время в проекте работают всего три взрослых специалиста (остальные – студенты), трудно рассчитывать, что в университетском проекте удастся полностью достичь обеих целей.

Тем не менее, в настоящее время участники проекта [26], по-видимому, обладают самым большим опытом в обрасти разработки СУБД на основе SCM. Совершенно необходимо начинать новые проекты, активно исследовать возможные подходы, проводить специальные семинары и конференции для обмена идеями и опытом.

В заключение этого подраздела замечу, что для меня очевидны потенциальные преимущества подхода СУБД на основе SCM для транзакционных приложений. Скорость обработки транзакций сможет сравняться со скоростью основной памяти, это принципиально новое качество. К сожалению, я не могу придумать сценарий, в котором от применения SCM можно получить значительные преимущества для аналитических приложений.

Считается признанной идея, что горизонтально масштабируемые аналитические СУБД нужно основывать на использовании массивно-параллельных архитектур и принципа shared nothing [28]. Современные аналитические базы данных настолько объемны, что только в кластере, узлы которых обладают весьма емкими средами хранения, можно полностью разместить базу данных. Даже при использовании дисковой памяти накладные расходы на пересылку данных по сети могут оказаться неприемлемыми. Если же в узлах используется SCM, то сетевые накладные расходы могут свести на нет все преимущества SCM.

3.2 SCM в объектно-ориентированных и XML-ориентированных СУБД

В 21-м веке объектно-ориентированные СУБД практически потеряли пользователей. При этом активно используются разнообразные средства объектно-реляционного отображения (Object-Relational Mapping, ORM), позволяющие объектно-ориентированным приложениям в объектной манере взаимодействовать с SQL-ориентированными базами данных [29]. На мой взгляд, в принципе для хранения объектов лучше было бы использовать ООСУБД, а не средства ORM[4].

Мне кажется, что распространенности ООСУБД во многом помешала свойственная им проблема, частично относящаяся к объектно-ориентированной модели данных [30]. Как известно, одним из основных понятий этой модели данных является объектный идентификатор (Object Identifier, OID), автоматически генерируемый системой при создании любого объекта, уникально отличающий этот объект от всех других объектов любого объектного типа и служащий своего рода абстрактным указателем на объект. В частности, с помощью OID’ов в модели ODMG образуются связи между объектами.

[4] Как демонстрируется в [30], с равным успехом можно использовать объектные возможности самого языка SQL, однако эта идея не получила широкого распространения.

При использовании в ООСУБД для хранения баз данных блочной внешней памяти затруднительно явно использовать в качестве OID обычные указатели. Кроме того, давно известна проблема преобразования OID'ов в обычные указатели при перемещении объектов из базы данных в объектно-ориентированную среду клиентских приложений [32]. Если основывать ООСУБД на SCM, обе проблемы, похоже, сильно упростятся, а навигационная природа ООСУБД не будет сильно тормозить ее работу, поскольку затраты на разыменование OID'ов можно свести практически к нулю.

Аналогично, использование SCM может возродить интерес к XML-ориентированным СУБД, в которых для поддержки путевых выражений и пр. приходится поддерживать массу ссылок, а для обеспечения более или менее приемлемой эффективности использовать изощренные схемы хранения [33]. Очевидно, что при наличии 64-разрядной адресации и достаточного объема основной энергонезависимой памяти XML-ориентированные СУБД можно резко упростить и ускорить.

4. Заключение

Как видно, сценариев, в которых SCM может значительно повысить эффективность СУБД и упростить их организацию, более чем достаточно. Нужно продолжать анализировать разные ветви дисциплины управления данными, чтобы не упустить других благоприятных возможностей применения SCM. Лично для меня было бы очень интересно найти пути использования SCM в аналитических СУБД. И конечно, требуется большое число исследовательских проектов, чтобы найти правильные пути разработки СУБД на основе SCM.

Список литературы

1. R. Bayer, E. McCreight. Organization and Maintenance of Large Ordered Indexes, Acta Informatica, vol. 1, issue 3, pp. 173–189, 1972
2. Joseph M. Hellerstein and Michael Stonebraker. Anatomy of a Database System. In Readings in Database Systems, 4th Edition. MIT Press, 2005, pp. 42-95
3. С.Д. Кузнецов. Базы данных. Академия, Серия: Университетский учебник, 2012 г., 496 стр.
4. P. Griffiths Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, T.G. Price. Access Path Selection in a Relational Database Management System. In Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, pp. 23-34
5. David J. DeWitt, Paula B. Hawthorn. A Performance Evaluation of Data Base Machine Architectures (Invited Paper). In Proceedings of the 7th International. Conference on Very Large Data Bases, 1981, pp. 199-214
6. David DeWitt, Jim Gray. Parallel database systems: the future of high performance database systems. Communications of the ACM, vol. 35, Issue 6, June 1992, pp. 85-98
7. David J. DeWitt, Randy H. Katz, Frank Olken. Leonard D Shapiro, Michael R. Stonebraker, David A. Wood. Implementation techniques for main memory database systems. In Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp. 1-8
8. Д.А. Шапоренков. Эффективные методы индексирования данных и выполнения запросов в системах управления базами данных в основной памяти. Диссертация на соискание ученой степени кандидата физико-математических. Санкт-Петербургский государственный университет. 2006
9. Tirthankar Lahiri, Marie-Anne Neimat and Steve Folkman. Oracle TimesTen: An In-Memory Database for Enterprise Applications. Bulletin of the Technical Committee on Data Engineering, vol. 36, no. 2, June 2013, pp. 6-13
10. Jan Lindstrцm, Vilho Raatikka, Jarmo Ruuth, Petri Soini, and Katriina Vakkila, IBM

solidDB: In-Memory Database Optimized for Extreme Speed and Availability. Bulletin of the Technical Committee on Data Engineering, vol. 36, no. 2, June 2013, pp. 14-20
11. Michael Stonebraker and Ariel Weisberg. The VoltDB Main Memory DBMS. Bulletin of the Technical Committee on Data Engineering, vol. 36, no. 2, June 2013, pp. 21-27
12. С.Д, Кузнецов. Транзакционные параллельные СУБД: новая волна. Труды ИСП РАН, т. 20, 2011, стр. 189-251
13. Novotný R., Kadlec J. and Kuchta R. NAND Flash Memory Organization and Operations. Journal of Information Technology & Software Engineering, vol. 5, issue 1, 2015. 8 p.
14. Сайт проекта FlashyDB, http://dblab.reutlingen-university.de/FDB.html. Data Management Lab, Reutlingen University, Germany. Дата обращения 10 октября 2017 г.
15. Ilia Petrov, Robert Gottstein, Sergej Hardock. DBMS on modern storage hardware. In Proceedings of the 31st International Conference on Data Engineering (ICDE), 2015, pp. 1545-1548
16. С.Д. Кузнецов, А.А. Прохоров. Алгоритмы управления буферным пулом СУБД при работе с флэш-накопителями. Труды ИСП РАН, т. 23, 2012, стр. 173-194. DOI: 10.15514/ISPRAS-2012-23-11
17. S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam. Phase-change random access memory: A scalable technology. Journal of Research and Development, vol. 52, No 4/5, 2008, pp. 465-479
18. D.B. Strukov, G.S. Snider, D.R. Stewart and R.S. Williams. The missing memristor found. Nature, 453, 1 May 2008, pp. 80-83
19. MRAM: Создание производства магниторезистивной оперативной памяти в России, http://www.rusnano.com/projects/portfolio/crocus-technology. Дата обращения 10 октября 2017 г.
20. Yiling Lin, IJessie Shen, DIGITIMES [Tuesday 26 September 2017]. Samsung ready to mass produce MRAM chips using 28nm FD-SOI process. https://digitimes.com/news/a20170925PD206.html. Дата обращения 10 октября 2017 г.
21. Joy Arulraj, Andrew Pavlo. How to Build a Non-Volatile Memory Database Management System. In Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1753-1758, 2017
22. Intel 64 and IA-32 Architectures. Software Developer's Manual. Documentation Changes. July 2017. https://software.intel.com/sites/default/files/managed/3e/79/252046-sdm-change-document.pdf. Дата обращения 10 октября 2017 г.
23. Альфред Ахо, Джон Хопкрофт, Джеффри Ульман, Структуры данных и алгоритмы. Вильямс, 2016, 400 стр.
24. С.Д. Кузнецов. Методы сортировки и поиска. http://citforum.ru/programming/theory/sorting/sorting2.shtml. 2003 г. Дата обращения 10 октября 2017 г.
25. Michael Stonebraker. The Design of the POSTGRES Storage System. In Proceedings of 13th International Conference on Very Large Data Bases, 1987, pp. 289-300
26. Сайт проекта Peloton: The Self-Driving Database Management System, http://pelotondb.io/. Database Group, Carnegie Mellon University. Дата обращения 10 октября 2017 г.
27. Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C. Mowry, Matthew Perron, Ian Quah, Siddharth Santurkar, Anthony Tomasic, Skye Toor, Dana Van Aken, Ziqi Wang, Yingjun Wu, Ran Xian, Tieying Zhang. Self-Driving Database Management Systems. In Proceedings of the 8th Biennial Conference on Innovative Data Systems Research (CIDR '17), Online Proceedings, 6 p.
28. С.Д. Кузнецов. К свободе от проблемы больших данных. Открытые системы, N 2, 2012, стр. 22-24
29. Ted Neward, The Vietnam of Computer Science. Ted Neward's Blog, Jun 26, 2006. Дата

обращения 10 октября 2017 г.
30. С.Д. Кузнецов. Объектные модели ODMG и SQL десять лет спустя: нет противоречий. Труды ИСП РАН, том 27, выпуск 1, 2015 г., стр. 173-192. DOI: 10.15514/ISPRAS-2015-27(1)-9
31. The Object Data Standard: ODMG 3.0. Edited by R.G.G. Cattel, Douglas K. Barry. Morgan Kauffmann Publishers, 2000, 280 p.
32. Alfons Kemper, Donald Kossmann. Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis. The VLDB Journal, vol. 4, issue 3, July 1995, pp 519–566
33. Ilya Taranov, Ivan Shcheklein, Alexander Kalinin, Leonid Novak, Sergei Kuznetsov, Roman Pastukhov, Alexander Boldakov, Denis Turdakov, Konstantin Antipin, Andrey Fomichev, Peter Pleshachkov, Pavel Velikhov, Nikolai Zavaritski, Maxim Grinev, Maria Grineva, Dmitry Lizorkin. Sedna: Native XML Database Management System (Internals Overview). In Proceedings of the 2010 International Conference on Management of Data, pp. 1037-1046.

Prospects and problems of using nonvolatile memory

S.D. Kuznetsov <kuzloc@ispras.ru>
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia.
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
Moscow Institute of Physics and Technology (State University),
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
National Research University Higher School of Economics (HSE)
11 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. At the beginning of the paper, it is demonstrated that the technology of the most widely used SQL-oriented DBMS is inextricably linked with HDD technology. Features of HDD affect the data structures and algorithms for performing operations, methods of managing the buffer pool of the DBMS, transaction management, query optimization, etc. An alternative to a disk DBMS is an in-memory DBMS, storing databases entirely in the main memory. Despite the fact that in-memory DBMS has a number of advantages over disk DBMS, at present there is practically no competition. This, first of all, is due to natural limitations on the size of databases, inherent in in-memory DBMS. At present, new types of data storage hardware have appeared: SSD – block solid-state drives and SCM – storage-class memory (non-volatile main memory). SSD characteristics made it expedient to develop a DBMS in terms of their exclusive use, but so far such a DBMS has not been created, and SSDs are used simply instead of HDDs in DBMS that do not take into account their features. The availability of SCM allows to radically simplify the architecture of the database and significantly improve their performance. To do this, you need to review many of the ideas used in disk-based databases.

Keywords: SQL-oriented DBMS; hard disk drive; cost-based query optimization; in-memory DBMS; solid-state drive; storage-class memory