DOI: 10.15514/ISPRAS–2021–33(3)–13

Аналитика в реальном времени, гибридная транзакционная/аналитическая обработка, управление данными в основной памяти и энергонезависимая память

[1,2,3,4,5] *С.Д. Кузнецов, ORCID: 0000-0002-8257-028X <kuzloc@ispras.ru>*
[6] *П.Е. Велихов, ORCID: 0000-0002-0644-8047 <pavel.velikhov@huawei.com>*
[6] *Ц. Фу, ORCID: 0000-0003-0244-1718 <fqiang.fuqiang@huawei.com>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, Москва, ул. А. Солженицына, д. 25*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*
[3] *Московский физико-технический институт, 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9*
[4] *НИУ «Высшая школа экономики», 101978, Россия, Москва, ул. Мясницкая, д. 20*
[5] *Российский экономический университет имени Г.В. Плеханова, 117997, Москва, Стремянный пер., 36*
[6] *Техкомпания Хуавэй. 121614, Россия, Москва, ул. Крылатская, д. 17, к. 2*

Аннотация. В наши дни аналитика в реальном времени – одно из наиболее часто используемых понятий в мире баз данных. В широком смысле этот термин означает очень быструю аналитику очень свежих данных. Обычно этот термин используется вместе с другими популярными терминами – *гибридной транзакционной / аналитической обработкой (HTAP)* и *обработкой данных в основной памяти*. Причина в том, что самый простой способ предоставить свежие оперативные данные для анализа – это объединить в одной системе как транзакционную, так и аналитическую обработку. Самый эффективный способ обеспечить быструю транзакционную и аналитическую обработку – хранить всю базу данных в основной памяти. Итак, с одной стороны, эти три термина связаны, но с другой стороны, каждый из них имеет собственное право на жизнь. В этой статье мы даем обзор нескольких систем управления данными в памяти, которые не являются системами HTAP. Некоторые из них являются чисто транзакционными, некоторые – чисто аналитическими, а некоторые поддерживают аналитику в реальном времени. Затем мы рассмотрим девять HTAP-СУБД с хранением баз данных в основной памяти, некоторые из которых не поддерживают аналитику в реальном времени. Существующие HTAP-СУБД реального времени с хранением баз данных в основной памяти имеют очень разнообразную и интересную архитектуру, хотя они используют ряд общих подходов: многоверсионное управление параллелизмом, многоядерное распараллеливание, расширенная оптимизация запросов, своевременная компиляция запросов и т.д. Кроме того, нас интересует, используют ли эти системы энергонезависимую память, и если да, то каким образом. Мы пришли к выводу, что появление нового поколения NVM будет значительно стимулировать использование энергонезависимой основной памяти в системах HTAP с хранением баз данных в основной памяти.

Ключевые слова: аналитика в реальном времени; гибридная транзакционная/аналитическая обработка; обработка данных в основной памяти; энергонезависимая память.

Для цитирования: Кузнецов С.Д., Велихов П.Е., Фу Ц. Аналитика в реальном времени, гибридная транзакционная/аналитическая обработка, управление данными в основной памяти и энергонезависимая память. Труды ИСП РАН, том 33, вып. 3, 2021 г., стр. 171-198. DOI: 10.15514/ISPRAS–2021–33(3)–13

Real-Time Analytics, Hybrid Transactional/Analytical Processing, In-Memory Data Management, and Non-Volatile Memory

[1,2,3,4,5] *S.D. Kuznetsov, ORCID: 0000-0002-8257-028X <kuzloc@ispras.ru>*
[6] *P.E. Velikhov, ORCID: 0000-0002-0644-8047 <pavel.velikhov@huawei.com>*
[6] *Q. Fu, ORCID: 0000-0003-0244-1718 <fqiang.fuqiang@huawei.com>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*
[3] *Moscow Institute of Physics and Technology (State University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia*
[4] *National Research University, Higher School of Economics 20, Myasnitskaya Ulitsa, Moscow, 101978, Russia*
[5] *Plekhanov Russian University of Economics, 36, Stremyanny lane, Moscow, 117997, Russia*
[6] *Huawei Technologies Co., Ltd., 17, building 2, st. Krylatskaya, Moscow, 121614, Russia*

Abstract. These days, *real-time analytics* is one of the most often used notions in the world of databases. Broadly, this term means very fast analytics over very fresh data. Usually the term comes together with other popular terms, *hybrid transactional/analytical processing (HTAP)* and *in-memory data processing*. The reason is that the simplest way to provide fresh operational data for analysis is to combine in one system both transactional and analytical processing. The most effective way to provide fast transactional and analytical processing is to store an entire database in memory. So on the one hand, these three terms are related but on the other hand, each of them has its own right to life. In this paper, we provide an overview of several in-memory data management systems that are not HTAP systems. Some of them are purely transactional, some are purely analytical, and some support real-time analytics. Then we overview nine in-memory HTAP DBMSs, some of which don't support real-time analytics. Existing real-time in-memory HTAP DBMSs have very diverse and interesting architectures although they use a number of common approaches: multiversion concurrency control, multicore parallelization, advanced query optimization, just in time compilation, etc. Additionally, we are interested whether these systems use non-volatile memory, and, if yes, in what manner. We conclude that an emergence of new generation of NVM will greatly stimulate its use in in-memory HTAP systems.

Keywords: real-time analytics; hybrid transactional/analytical processing; in-memory data processing; non-volatile memory

For citation: Kuznetsov S.D., Velikhov P.E., Fu Q. Real-Time Analytics, Hybrid Transactional/Analytical Processing, In-Memory Data Management, and Non-Volatile Memory. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 3, 2021, pp. 171-198 (in Russian). DOI: 10.15514/ISPRAS–2021–33(3)–13

1. Введение

Термины и концепции в области управления данными постоянно меняются, и аналитика в реальном времени в настоящее время является одной из самых популярных концепций. В целом, аналитика в реальном времени предполагает *быструю* аналитическую обработку *свежих* данных. Оба ключевых слова в приведенном выше предложении, *быстрая обработка* и *свежие данные*, не имеют абсолютного смысла. Аналитическая обработка в режиме реального времени должна происходить настолько быстро, насколько этого требуют

клиенты (предприятия), а данные могут быть настолько свежими, насколько они могут быть предоставлены базовой системой управления данными.

В любом случае, наиболее естественным источником свежих корпоративных данных являются данные, генерируемые транзакциями того же предприятия. Обычно все корпоративные транзакции обрабатываются некоторыми СУБД, ориентированными на OLTP, а корпоративная аналитика поддерживается некоторыми СУБД, ориентированными на OLAP. Таким образом, для обеспечения аналитики в реальном времени необходимо очень быстро передавать данные из хранилища транзакционной системы в хранилище аналитической системы. Другими словами, чтобы обеспечить аналитику свежих транзакционных данных, необходимо предоставить очень быстрый механизм ETL (извлечение-преобразование-загрузка). Но никто не знает, как реализовать такой быстрый ETL, и для того, чтобы сделать аналитику в реальном времени возможной, была введена концепция *гибридной транзакционной/аналитической обработки* (hybrid transactional/analytical processing, HTAP).

В широком смысле *HTAP* означает, что мы доставляем свежие транзакционные данные для аналитической обработки без какого-либо ETL посредством тесной интеграции транзакционных и аналитических хранилищ данных. Чтобы добиться такой интеграции магазинов, необходимо также интегрировать транзакционные и аналитические подсистемы. Легко видеть, что такая интеграция была естественной до начала эры «один размер не подходит для всех» [1], потому что универсальные СУБД полностью поддерживали стандарт SQL и теоретически могли поддерживать смешанные рабочие нагрузки OLTP / OLAP.

Основным преимуществом специализированных СУБД было использование структур данных и алгоритмов, наиболее подходящих для соответствующих видов рабочей нагрузки. Например, специализированные аналитические СУБД, такие как Vertica [2], используют разделенные хранилища данных с поколоночным хранением таблиц и оптимизированы для обработки сложных аналитических запросов с несколькими соединениями, в то время как специализированные транзакционные СУБД, такие как VoltDB [3], поддерживают хранилище данных в памяти на основе строк и являются оптимизированными для обработки коротких и простых транзакций. Безусловно, специализированная транзакционная СУБД выигрывает у универсальной СУБД, когда рабочая нагрузка является транзакционной, и специализированная аналитическая СУБД выигрывает, когда рабочая нагрузка является аналитической.

Поэтому «идеалистическая» цель подхода HTAP – одновременно обеспечить в рамках одной системы функциональность и производительность специализированных OLTP- и OLAP-СУБД – кажется недостижимой. Прагматическая цель этого подхода – создание системы, которая обеспечивает разумную пропускную способность для транзакционных рабочих нагрузок и одновременно аналитику в реальном времени по достаточно свежим данным - несомненно достижима (как мы покажем в этой статье).

Очевидно, что требования разумной пропускной способности для транзакционных рабочих нагрузок и свежести данных для аналитических запросов противоречат друг другу. Чтобы обеспечить максимальную актуальность данных, необходимо сделать доступными для анализа все данные, генерируемые текущими обрабатываемыми транзакциями. Однако в этом случае транзакции будут конкурировать с аналитическими запросами, и обработка транзакций станет медленнее. Все известные системы HTAP используют тот или иной компромисс для разумного (в некоторой степени) удовлетворения этих противоречивых требований путем разделения аналитической и транзакционной частей базы данных и, возможно, преобразования данных из представления, хорошо подходящего для обработки транзакций (обычно таблицы с хранением по строкам) в более подходящее для аналитики представление (обычно поколоночное представление таблиц). (Если хотите, это можно назвать облегченным ETL.)

Другое измерение – это носители для хранения данных. Большинство современных СУБД категории HTAP оптимизированы для хранения данных в основной памяти. Это означает, что все структуры данных и алгоритмы, используемые в таких системах, разработаны в соответствии с предположением, что вся обработка данных будет выполняться в основной памяти без ввода-вывода с использованием внешних запоминающих устройств. Вообще говоря, концепция СУБД с хранением баз данных в основной памяти (in-memory СУБД) не нова. Согласно [4], первая попытка реализовать систему базы данных, которая хранит все данные в памяти, была предпринята IBM в 1976 году (IMS Fast Path). Новая волна in-memory СУБД с оперативной памятью наблюдалась в 1990-х годах. Тогда и позже (в начале 2000-х) такие СУБД в основном специализировались на обработке транзакций. Сейчас почти все существующие HTAP-СУБД в той или иной степени относятся к классу систем in-memory. Хранение всех данных (или, по крайней мере, *горячих* данных) в основной памяти обеспечивает более высокую скорость обработки транзакций, что частично сглаживает негативные эффекты одновременно выполняемых аналитических запросов.

Таким образом, понятия аналитики в реальном времени, гибридной транзакционной/аналитической обработки и управления данными в основной памяти являются взаимосвязанными понятиями, хотя каждое из них имеет свой собственный смысл и право на жизнь. Мы обсудим более подробно их происхождение, собственное значение и взаимосвязь в разд. 2.

Еще одна концепция, последняя, но не менее важная для целей этой статьи, – это *энергонезависимая основная память* (non-volatile main, NVM), которая также называется *постоянной памятью* (persistent memory), *памятью класса хранения данных* (storage-class memory) и т.д. NVM на самом деле является обычной памятью с байтовой адресацией (как традиционная RAM), но сохраняет свое состояние после отключения питания. Эти особенности NVM позволяют использовать ее как одноуровневую среду хранения данных в системах управления данными.

NVM наиболее хорошо подходит для чисто транзакционных СУБД, работающих на многоядерных компьютерах [5]. В этом случае основным преимуществом использования NVM является то, что важная функция долговечности транзакций будет обеспечиваться без какой-либо журнализации во внешней памяти. Таким образом, транзакции, обновляющие базу данных, будут обрабатываться с той же скоростью, что и транзакции только для чтения, а общая пропускная способность для транзакционных рабочих нагрузок будет намного выше, чем у традиционных транзакционных in-memory СУБД (эти системы должны журнализовать все операции обновления в энергонезависимой внешней памяти, чтобы обеспечить долговечность транзакций).

Однако в настоящее время наблюдается тенденция к объединению в одной и той же системе баз данных функций обработки транзакций и аналитики в реальном времени, то есть к использованию подхода HTAP. Поэтому представляется интересным и полезным проанализировать, используют ли разработчики имеющихся в настоящее время in-memory СУБД категории HTAP (или собираются ли они использовать) NVM, и, если да, то как они его используют (или планируют использовать).

Основной вклад статьи этой статьи состоит в следующем:

- предоставляются более или менее точные определения и объяснения связанных, но различных понятий аналитики в реальном времени, гибридной транзакционной / аналитической обработки и управления данными в основной памяти;
- обоснована принципиальная важность энергонезависимой памяти в будущих СУБД категории HTAP;
- кратко описан общий ландшафт существующих систем управления данными в основной памяти;
- дан обзор новейших in-memory HTAP-СУБД и использования в них NVM.

Оставшаяся часть статьи имеет следующую структуру. В разд. 2 предлагается некоторая предыстория, включая определения и объяснения терминов и понятий, используемых в статье. В разд. 3 мы приводим общую классификацию современных in-memory СУБД и кратко характеризуем некоторые системы, не принадлежащие к категории HTAP. В разд. 4 более подробно описаны архитектурные и основные функциональные особенности основных коммерческих и академических разработок in-memory HTAP-СУБД. Для каждой системы мы указываем, используется ли в ней NVM, и, если да, то каким образом. Разд. 5 содержит некоторые заключительные соображения авторов и завершает статью.

2. Предпосылки: определения, объяснения и обсуждение

Имеются три концепции и соответствующие термины, которые обычно объединяются в области баз данных, но каждое из них имеет собственное значение и фундаментальное право на отдельную жизнь. Эти термины таковы:

- аналитика в реальном времени;
- гибридная транзакционная / аналитическая обработка (HTAP); а также
- управление данными в памяти.

Мы начнем с рассмотрения смысла каждого из этих терминов, а затем обсудим, почему они в настоящее время объединяются. Наконец, в этом разделе мы также кратко рассмотрим понятие энергонезависимой памяти, текущее состояние соответствующей технологии и ее связь с упомянутыми связанными концепциями.

2.1 Аналитика в реальном времени

Согласно Глоссарию Gartner [6], «Аналитика в реальном времени – это дисциплина, к которой применяется логика и математика к данным, чтобы обеспечить их понимание для быстрого принятия лучших решений. В некоторых случаях использование реального времени просто означает, что аналитика завершается в течение нескольких секунд или минут после поступления новых данных».

Первая (общая) часть этого определения фактически означает, что аналитические инструменты в реальном времени должны удовлетворять потребности лиц, принимающих бизнес-решения. Кажется, это общая цель любого поставщика, предоставляющего аналитические решения. В частности, для достижения этой цели помогают хранение таблиц по столбцам и различные виды разделения базы данных, используемые в специализированных аналитических СУБД, таких как, например, Teradata [7] или Vertica [2].

Второе утверждение в приведенной выше цитате дает более близкую к нашему пониманию, более конкретную и слегка противоречивую интерпретацию концепции аналитики в реальном времени. На самом деле это означает, что:

- данные должны быть доступны для анализа после их создания *как можно скорее*, и
- анализ данных должен быть завершен *как можно быстрее*.

Во-первых, обратите внимание на наличие двух экземпляров оборота «как можно». Для завершения аналитики существует большой разрыв между несколькими секундами и несколькими минутами. Итак, мы видим совершенно особую интерпретацию смысла реального времени: время, потраченное на аналитику, должно быть настолько реальным, насколько это возможно.

Во-вторых, для обеспечения свежести данных, вероятно, нужно отказаться от всей предварительной подготовки данных к анализу. Таким образом, анализ должен проводиться над данными, представленными в их первоначальном представлении (возможно, с небольшим лексическим преобразованием, например, из строковой формы в форму, основанную на столбцах). Непросто обеспечить быстрое выполнение сложных аналитических запросов к неподготовленным данным.

И, в-третьих, приведенное выше определение ничего не говорит о способах достижения цели предоставления аналитики в реальном времени и не указывает каких-либо точных характеристик «реального времени». С одной стороны, такое неопределенное «определение» облегчает разработку системы, удовлетворяющей «требованиям» аналитики в реальном времени. Но с другой стороны, оно делает практически невозможным сравнение любых двух систем этой категории.

2.2 Гибридная транзакционная/аналитическая обработка

Как говорится в Википедии со ссылкой на Gartner [8], «гибридная транзакционная / аналитическая обработка (HTAP) – это новая архитектура приложений, которая “ломает стену” между обработкой транзакций и аналитикой. Это позволяет принимать более информированные решения в режиме реального времени». В соответствии с этим определением стоит сделать несколько комментариев.

(a) Любая универсальная реляционная СУБД, корректно поддерживающая стандарт SQL, относится к категории HTAP в смысле первой части этого определения. Действительно, стандарт SQL предписывает поддержку транзакций ACID со всеми обычными гарантиями. Следовательно, любая СУБД, поддерживающая стандарт SQL, должна быть способна обрабатывать любую транзакционную рабочую нагрузку. Стандарт SQL предусматривает поддержку произвольно сложных аналитических запросов, а также наличие достаточно богатого и расширяемого набора аналитических операций. Таким образом, любая СУБД, поддерживающая стандарт SQL, потенциально должна быть способна обрабатывать любую аналитическую рабочую нагрузку. В определении ничего не говорится о производительности такой обработки.

(b) Фактическая «стена» между OLTP и OLAP появилась только после появления концепции Майкла Стоунбрейкера «один размер не подходит для всех» [1]. Специализированные транзакционные и аналитические СУБД либо не поддерживают весь стандарт SQL, либо поддерживают транзакционные и аналитические функции SQL с неравной производительностью.

(c) Подход HTAP можно рассматривать как шаг назад от специализированных систем баз данных. Одним из основных пунктов критики Стоунбрейкера универсальных СУБД была их сложность. Специализированные DMBS, естественно, намного проще универсальных, поскольку они предоставляют лишь ограниченную функциональность. Большинство усилий сосредоточено на обеспечении производительности. При разработке HTAP-СУБД нам необходимо одновременно обеспечить гибридную функциональность и сохранить (или хотя бы попытаться сохранить) производительность специализированных систем баз данных.

(d) Подход HTAP в смысле приведенного выше определения может помочь решить только первую часть проблемы аналитики в реальном времени – предоставить аналитический доступ к свежим данным. Высокая производительность аналитической обработки может быть достигнута за счет использования оптимизированной архитектуры СУБД категории HTAP, методов взаимодействия транзакционных и аналитических движков внутри системы, а также оптимизации аналитических запросов.

В завершение этого подраздела приведем еще одну цитату [9], которая отражает точку зрения Хассо Платтнера (Hasso Plattner), соучредителя SAP, основателя института Хассо Платтнера (Hasso-Plattner-Institut für Digital Engineering, HPI) и главного инициатора HTAP-СУБД HANA в SAP. Авторы [9] писали, что «Использование общей модели базы данных для корпоративных систем может значительно улучшить эксплуатационные качества и снизить сложность использования систем, обеспечивая при этом доступ к данным в реальном времени

и совершенно новые возможности». Кажется, что главная мысль здесь – *эксплуатационные качества*. Очевидно, что если предприятие может использовать одну гибридную базу данных вместо как минимум двух отдельных транзакционных и аналитических баз данных, это будет намного проще и дешевле.

2.3 Управление данными в памяти

В глоссарии Gartner содержится следующее утверждение о HTAP и управлении данными в памяти: «Архитектура гибридной транзакции/аналитической обработки (HTAP) лучше всего обеспечивается методами и технологиями вычислений в основной памяти (in-memory computing, IMC), позволяющими выполнять аналитическую обработку над тем же хранилищем данных (в основной памяти), которое используется для обработки транзакций».

Вероятно, это правда, и в разд. 4 мы увидим несколько примеров, подтверждающих это утверждение. Однако концепции in-memory и HTAP – это не одно и то же. In-memory-СУБД появились намного раньше, чем понятие HTAP, и были ориентированы на очень быструю обработку транзакций. Очень хороший обзор истории и текущего состояния технологии баз данных в основной памяти представлен в статье [4], написанной руководителями нескольких проектов in-memory СУБД.

Даже сейчас существует несколько чисто транзакционных СУБД в оперативной памяти. Некоторые из них являются успешными коммерческими продуктами, такими как VoltDB [3] и SolidDB [10], а другие представляют собой академические прототипы, например, Silo [11]. Среди in-memory СУБД есть и чисто аналитические системы управления данными, например, традиционная массивно-параллельная аналитическая СУБД Exasol [12], многомерные специализированные системы Cognos TM1 [13] и Essbase [14], а также улучшенная версия Spark-PMoF [15] распределенной аналитической среды Spark. Мы кратко обсудим эти системы в следующем разделе.

Как мы упоминали во введении, хранение базы данных полностью в основной памяти в принципе позволяет системе баз данных HTAP сократить время выполнения как транзакций, так и аналитических запросов. Однако для обеспечения быстрой аналитики над свежими данными необходимо также сократить задержку, с которой транзакционные данные становятся доступными для аналитической обработки. Это можно сделать, например, используя для обработки транзакций не традиционное представление таблицы на основе строк, а представление таблицы в виде столбцов, которое лучше всего подходит для аналитической обработки данных.

Это была одна из основных идей Хассо Платтнера при разработке HANA. Как он писал в [16], «Ранние тесты в SAP и HPI с базами данных в основной памяти реляционного типа, основанными на хранении таблиц по строкам, не показали значительных преимуществ перед ведущими СУБД с эквивалентной памятью для кэширования. Здесь родилась альтернативная идея – изучить преимущества использования для OLTP баз данных с поколоночным хранением».

Они сделали это, и HANA в своем обычном, наиболее оптимизированном режиме может выполнять параллельную обработку обоих видов рабочей нагрузки в in-memory хранилище поколоночных таблиц. Однако стоит отметить, что почти все существующие СУБД категории HTAP (включая HANA) поддерживают хранилища таблиц по столбцам и строкам в одной базе данных. Похоже, что хранилище по строкам по-прежнему необходимо, если требуется максимальная пропускная способность для транзакционных рабочих нагрузок.

Наконец, заметим, что все существующие СУБД в памяти, поддерживающие SQL, имеют ряд общих, очень важных технических характеристик, таких как своевременная компиляция запросов в оптимизированный машинный код, агрессивное сжатие данных, активное использование инструкций SIMD процессора, использование структур данных и алгоритмов без блокировок и защелок, многоверсионное управление параллелизмом и т.д. Однако мы

почти не будем касаться этих вопросов далее в этой статье, потому что здесь нас больше всего интересуют общие архитектуры и принципы хранения данных в in-memory HTAP-СУБД.

2.4 Энергонезависимая память и ее правильное место в технологии баз данных

Откровенно говоря, основной интерес авторов этой статьи – возможное включение поддержки новой энергонезависимой памяти (NVM) (также называемой постоянной памятью (PMEM) и памятью класса хранения (SCM)) в архитектуры современных (в основном класса in-memory) СУБД. Мы не будем приводить здесь обзор текущего состояния технологии NVM, который можно найти где-либо еще (например, в [5]). Отметим только со ссылкой на [5], что в настоящее время практически доступен только один вид NVM в форм-факторе DIMM – Intel Optane DIMM. Общие характеристики этого продукта – большая задержка (на несколько двоичных порядков выше, чем у DRAM), большая задержка операций записи, чем чтения, и большая емкость (до 512 Гб на модуль). Электронная промышленность обещает предоставить (надеемся, в ближайшее время) следующие поколения NVM, которые будут иметь характеристики задержки и емкости не хуже, чем у текущих DRAM, и характеристики выносливости не хуже, чем у существующих HDD (или, по крайней мере, SSD).

Как уже упоминалось, лучшее место для NVM в области технологий баз данных – это транзакционная in-memory (фактически, in-NVM) СУБД, работающая на многоядерном сервере. Энергонезависимость NVM позволяет использовать одноуровневую архитектуру хранения данных вообще без использования каких-либо внешних запоминающих устройств. Напротив, текущие транзакционные in-memory СУБД вынуждены использовать постоянное внешнее хранилище, чтобы обеспечить важную характеристику долговечности зафиксированных транзакций. Кроме того, одноуровневая архитектура позволяет значительно упростить структуры данных и алгоритмы, используемые для управления базами данных. В такой системе любая транзакция полностью обрабатывается в одном потоке, строго соответствующем ядру процессора. Следовательно, если конкуренция за объекты базы данных между транзакциями рабочей нагрузки не очень высока (транзакции конфликтуют редко), производительность системы может расти почти линейно с увеличением количества ядер.

Однако очевидной текущей тенденцией является подход HTAP с управлением базами данных в основной памяти. Соответствующие системные архитектуры намного сложнее по сравнению с чисто транзакционными системами. Одноуровневое хранилище может иметь недостаточный размер. Поэтому цель данной статьи – проанализировать основные принципы архитектурной организации современных HTAP-СУБД, понять, может ли быть полезной для них энергонезависимая основная память и, если да, то как ее лучше всего использовать. Последний пункт особенно непонятен в отношении аналитики в реальном времени.

Но прежде чем приступить к этому анализу, мы кратко опишем общий ландшафт систем управления данными с хранением баз данных в основной памяти.

3. Пейзаж систем управления данными в памяти

На рис. 1 представлена репрезентативная выборка систем управления данными с хранением баз данных в основной памяти. Точнее, почти все системы в этой таблице, кроме одной (Spark-PMoF), на самом деле являются СУБД разных классов. Три класса здесь включают примеры чисто транзакционных СУБД (синий сектор), гибридных СУБД, поддерживающих как транзакционную, так и аналитическую рабочую нагрузку (темно-красный двойной сектор) и чисто аналитических систем управления данными (красный сектор).

Основное внимание в данной статье уделяется архитектурам HTAP, и мы дадим более подробный анализ решений этого «среднего» класса в следующем разделе. Сейчас мы кратко

рассмотрим представителей «крайних» классов – чисто транзакционные и чисто аналитические системы.

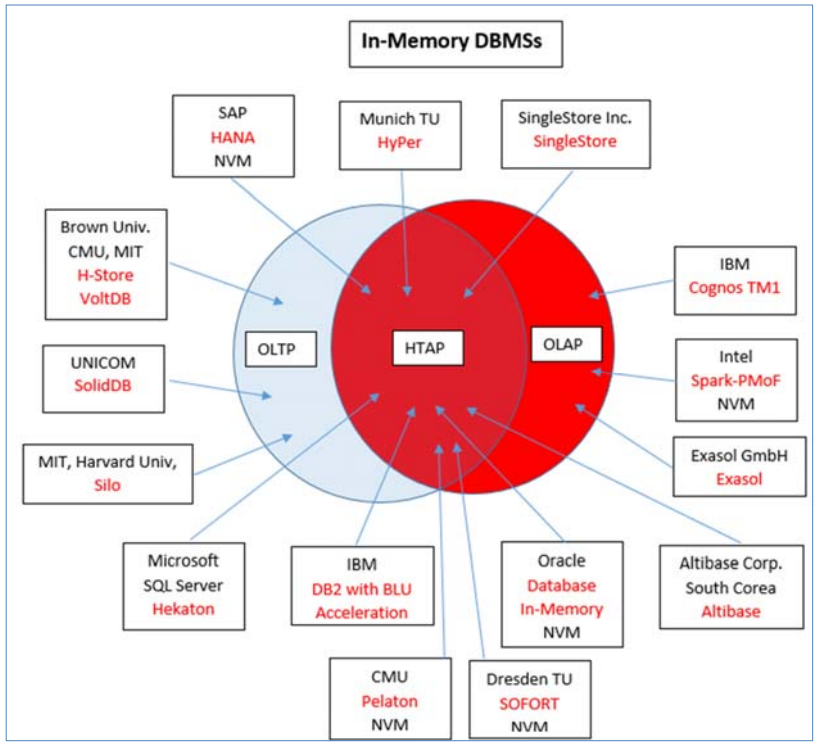


Рис. 1. Системы управления данными с хранением баз данных в основной памяти
Fig. 1. In-memory data management systems

3.1 Транзакционные in-memory СУБД

3.1.1 H-Store/VoltDB

Как мы уже упоминали, самая известная система первого класса – это VoltDB [3]. VoltDB является коммерческим преемником H-Store, академического проекта, инициированного Майклом Стоунбрейкером и реализованного консорциумом четырех крупных университетов США (Браун, Карнеги-Меллон, Массачусетский технологический институт и Йельский университет). Основные идеи H-Store изначально были опубликованы в [17].

Коротко говоря, H-Store/VoltDB – это массивно-параллельная, многораздельная, транзакционная in-memory СУБД с хранилищем таблиц на основе строк. Уникальные особенности этой системы включают отказ от ведения журнала транзакций. Система вообще не использует внешнее хранилище. И надежность транзакций, и надежность базы данных поддерживаются механизмом репликации.

Каждый узел системы выполняет локальные транзакции последовательно, одну за другой, чтобы исключить любую форму конкуренции локальных транзакций и, следовательно, избежать их локальных откатов. Разделение базы данных основано на статическом анализе

транзакций, участвующих в рабочей нагрузке (код всех транзакций должен быть доступен заранее). Цель этого анализа (а затем разделения) – минимизировать количество распределенных транзакций, для обработки которых требуются данные из более чем одного раздела.

В действительности система позволяет выполнять распределенные транзакции, которые фиксируются по традиционному двухфазному протоколу. Однако при обработке распределенных транзакций пропускная способность системы естественным образом снижается.

3.1.2 SolidDB

SolidDB [10, 18] – это продукт, который разрабатывался финской компанией Solid Information Technology с 1992 года. Затем он был приобретен IBM в 2007 году и продан компании UNICOM Global в 2014 году. В настоящее время SolidDB в основном используется для поддержки телекоммуникационных и сетевых приложений.

Система одновременно поддерживает два строковых хранилища данных: на дисках и в памяти. Дисковое хранилище полностью традиционное – данные хранятся в блоках дисков; индексы основаны на B + -деревьях. Для хранилища в основной памяти используются специальные, оптимизированные для основной памяти структуры данных. В частности, индексы для таблиц, резидентных в основной памяти, основаны на комбинации B-дерева и Patricia-trie. Решение о размещении таблицы (в памяти или на диске) должно приниматься при создании этой таблицы. Каждая операция запроса или обновления может относиться к таблицам как в памяти, так и на диске с полными транзакционными гарантиями.

Дополнительной особенностью СУБД SolidDB является возможность встраивать ее в приложения. В этом режиме экземпляр СУБД находится в адресном пространстве каждого такого приложения, а часть базы данных в основной памяти находится в сегменте совместно используемой памяти, который присоединен к этой виртуальной памяти. Детали реализации (например, как в этом случае поддерживается защита базы данных от прикладного программного обеспечения (и поддерживается ли она)) не предоставляются.

3.1.3 Silo

Проект Silo [11] был реализован в Гарвардском университете в начале 2010-х годов. Работой руководили лауреат премии Тьюринга Барбара Лисков (Barbara Liskov) и известный исследователь баз данных Сэмюэл Мэдден (Samuel Madden). Дизайн системы кажется предельно современным и интересным. К сожалению, проект мертв, хотя исходный код Silo все еще доступен на GitHub [19].

СУБД Silo ориентирована на транзакционное управление данными в основной памяти на современных многоядерных компьютерах. Хранение таблиц организовано с использованием первичных и вторичных индексов на основе оптимизированной для основной памяти разновидности B-дерева (с элементами префиксного дерева). Первичный индекс обеспечивает поиск кортежа по его первичному ключу; вторичный позволяет найти первичные ключи всех кортежей с заданным вторичным ключом.

Каждая транзакция полностью обрабатывается в одном потоке, выполняемом на отдельном ядре процессора. Все потоки имеют общий доступ ко всей памяти, занимаемой базой данных. Долговечность транзакций обеспечивается с помощью журнала транзакций, который хранится во внешнем постоянном хранилище. Чтобы обеспечить сериализуемость транзакций, Silo использует своего рода оптимистичный контроль параллелизма, разделяющий время на эпохи и использующий блокировки вместе с барьерной синхронизацией во время фиксации транзакции.

3.2 Аналитические системы с хранением баз данных в основной памяти

3.2.1 Exasol

Exasol [12] является основным продуктом немецкой компании Exasol GmbH, основанной в 2000 году. Вкратце, Exasol – это массивно-параллельная СУБД без совместного использования ресурсов (но с возможностью репликации) с хранением баз данных в основной памяти. Все аналитические запросы обычно обрабатываются распределенным образом на основе локальных поколоночных хранилищ таблиц каждого узла. Для обеспечения быстрой вставки новых данных система поддерживает дополнительное хранилище таблиц по строкам. Новые строки сливаются с основным поколоночным хранилищем в фоновом режиме.

Exasol использует ETL для добавления новых данных в базу данных. Источниками данных могут быть внешние транзакционные системы, Hadoop, веб-журналы и т.д. Во время локальной (внутри узла) обработки запроса (или части запроса) система активно использует симметричный параллелизм и инструкции процессора SIMD. Кроме того, система обрабатывает локальные запросы с активным использованием кеширования.

Стоит отметить, что Exasol демонстрирует лучшие результаты в бенчмарке TPC-H при масштабировании до сотен терабайт данных. Однако для справедливости следует сказать, что это единственная СУБД с хранением баз данных в основной памяти, которая участвует в этом соревновании.

3.2.2 Многомерные аналитические in-memory СУБД

В аналитических продуктах IBM и Oracle используются две хорошо известные системы многомерных баз данных в оперативной памяти, Cognos TM1 и Essbase. Эти СУБД кажутся очень похожими по всем характеристикам – они поддерживают многомерные кубы данных в основной памяти с предварительно вычисленными агрегатами, тесно интегрированы с электронными таблицами, такими как Excel, и т.д. Конечно, обе системы используют ETL для загрузки новых данных в базу данных. Еще одна общая черта – отсутствие какого-либо описания внутренней архитектуры системы, структур данных и алгоритмов. Поэтому ниже мы приводим лишь краткую историю этих продуктов.

Как гласит история [13], TM1 была впервые выпущена в начале 1983 года компанией Sinper Corporation, которая была приобретена Applix Inc. в 1996 году. Затем Applix была куплена канадской компанией Cognos в 2007 году, и, наконец, Cognos была приобретена IBM позже в том же году. В настоящее время аналитический сервер TM1 используется в более общем программном обеспечении IBM Planning Analytic.

Первая версия Essbase была разработана в 1992 году компанией Arbor Software. В 1998 году Arbor объединилась с компанией Hyperion Software, и Essbase начала использоваться в составе IBM DB2 OLAP Server, существовавшего до 2005 года. Компания Hyperion, в свою очередь, была приобретена Oracle в 2007 году. В настоящее время Essbase [14] является одним из ключевых компонентов Oracle Exalytics In-Memory Machine, специализированной аналитической комбинации аппаратного и программного обеспечения [20].

3.2.3 Spark PmoF

Популярная среда распределенной обработки данных Spark [21] на самом деле является оптимизированной версией MapReduce, которую можно грубо рассматривать как аналитическую систему в оперативной памяти. Однако при обработке сложных запросов с несколькими фазами map-reduce Spark демонстрирует значительное снижение производительности, поскольку перераспределение данных (перемешивание) выполняется с использованием HDFS (в реализации Apache), то есть через удаленное внешнее хранилище.

Инженеры Intel предложили (и реализовали) еще одну важную оптимизацию Spark, Spark-PMoF [15], используя Persistent Memory over Fabrics (PMoF), которая (опять же грубо говоря) является энергонезависимой памятью с байтовой адресацией, доступной через механизм RDMA (удаленный прямой доступ к памяти). Благодаря этой оптимизации при выполнении конкретного задания Spark не использует внешнее хранилище после начальной загрузки данных и выглядит как настоящая система управления и обработки данных в основной памяти. Наверное, стоит отметить, что это проект с открытым исходным кодом, а код и некоторая документация доступны на GitHub [22].

Приведенные в этом разделе примеры очень разных систем показывают, что ландшафт систем управления данными в памяти чрезвычайно разнообразен, и подкласс in-memory HTAP-СУБД является лишь частью этого ландшафта. Однако именно этот подкласс в данной статье представляет для нас основной интерес, и в следующем разделе мы переходим к обсуждению систем категории HTAP с хранением баз данных основной памяти.

4. In-memory HTAP-СУБД

Основная концепция подхода HTAP – объединение в одной системе баз данных возможностей как обработки транзакций, так и аналитики. Помимо этой общей характеристики, СУБД HTAP должна одновременно обеспечивать:

- очень быструю обработку транзакций;
- очень быстрое предоставление свежих данных, генерируемых транзакциями, для анализа данных; а также
- очень быструю обработку аналитических запросов произвольной сложности.

Одновременно достичь этих претенциозных целей нелегко. Однако есть несколько примеров СУБД, которые демонстрируют желаемые характеристики с использованием различных архитектурных и алгоритмических подходов, хотя все они используют хранилище данных в основной памяти.

В этом разделе мы кратко обсудим происхождение и основные архитектурные особенности этих систем. Мы начинаем с четырех систем основных поставщиков СУБД, а именно SAP, Microsoft, Oracle и IBM. Затем мы рассматриваем пару менее масштабных коммерческих СУБД и, наконец, представляем несколько прототипов академических систем. В каждом случае мы отмечаем, использует ли соответствующая система энергонезависимую память.

4.1 SAP HANA

HANA, аббревиатура от High-Performance Analytic Appliance, является основным продуктом управления базами данных компании SAP SE. Прежде всего, система используется в ERP-решениях самой SAP. Однако HANA приобретает все большую популярность как отдельный инструмент для разработки и поддержки приложений баз данных, которым требуется как транзакционная, так и аналитическая обработка данных. Система доступна в различных воплощениях, например, в кластерной или облачной версии. Однако здесь мы ограничимся основной конфигурацией HANA для одной машины.

4.1.1 Происхождение и предыстория

SAP SE была первой компанией в истории новейшего времени, которая решила объединить транзакционную и аналитическую обработку в одной системе, и Хассо Платтнер, как кажется, был отцом-основателем этого подхода. Еще в 2008 году Вишал Сикка (Vishal Sikka), который был техническим директором SAP с 2007 по 2014 год, написал в своем блоге: «… Хассо Платтнер вдохновил меня на проведение эксперимента, который мы назвали Hana… [и который] продемонстрировал, что возможна совершенно новая архитектура приложения,

обеспечивающая комплексную аналитику и агрегацию в реальном времени актуальных транзакционных данных…» [23].

Публикации, посвященные HANA, показывают, что главным архитектором системы с 2009 по 2019 год был Франц Фаербер (Franz Faerber). Активными участниками проекта были (и остаются) Вольфганг Ленер (Wolfgang Lehner) и члены возглавляемой им группы баз данных Дрезденского технического университета.

HANA не создавалась с нуля. На механизм хранения таблиц по столбцам в основной памяти повлияли собственные разработки SAP -- механизм текстового поиска TREX (2001) [24] и SAP Business Warehouse Accelerator (BWA, 2005) [25]. Механизм хранения таблиц по строкам в основной памяти основан на решении P*TIME [26], разработанном компанией Transact in Memory, Inc., которая было приобретено SAP в 2005 году. Механизм MaxDB [27], который был приобретен SAP в 1997 году у Software AG (тогда он назывался Adabas D), использовался для обеспечения уровня постоянного хранения данных. Первый релиз HANA был выпущен SAP в конце 2010 года.

4.1.2 Архитектурные особенности

Рис. 2 демонстрирует общую архитектуру HANA [28]. Мы обсуждаем только часть этой архитектуры, относящуюся к функциям HTAP в основной памяти.

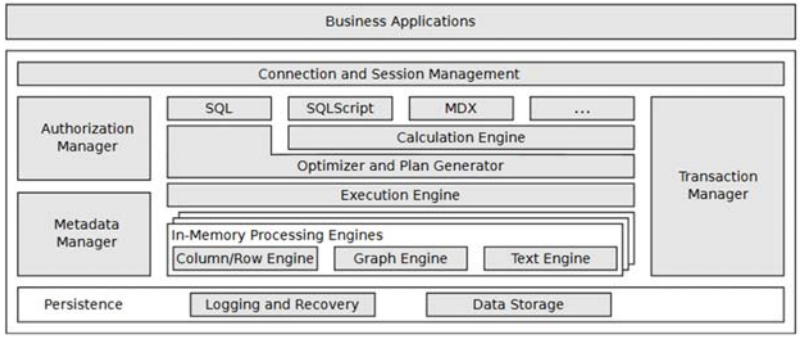


Рис. 2. Архитектура HANA [28]
Fig. 2. HANA architecture [28]

HANA позволяет использовать для одной и той же базы данных хранилища таблиц как по столбцам, так и по строкам. При создании новой таблицы пользователь может выбрать, как ее хранить, по строкам или по столбцам. Согласно первоначальным проектным решениям [16] предпочтительнее использовать хранилище по столбцам, и HANA оптимизирована для такой организации таблиц. Вероятно, хранилище столбцов не позволяет системе продемонстрировать высочайшую производительность обработки транзакций, потому что обновление хранилища столбцов – это операция, требующая много времени. Однако взамен этого поколоночный способ хранения таблиц базы данных снижает затраты на память из-за агрессивного сжатия данных. Чтобы достичь более высокой скорости обработки транзакций, можно использовать дорогостоящее хранилище строк.

Вообще говоря, каждый столбец таблицы хранится в хранилище столбцов с использованием двух структур данных – словаря и индексного вектора. Словарь содержит все различные значения, находящиеся в данный момент в столбце, а вектор индекса связывает каждую строку таблицы с соответствующей записью словаря. Для поддержки запросов по диапазону значений столбца, которые распространены в аналитике, HANA сохраняет отсортированный порядок элементов словаря. Однако это делает операции обновления, часто используемые при обработке транзакций, очень дорогими. Поэтому для каждого столбца поколоночной

таблицы имеются две части хранения: основная часть хранит словарь столбца в отсортированном порядке, а в дельта-части такой порядок не поддерживается. Все операции над таблицей используют как основные данные каждого столбца, так и его дельту. Основываясь на информации о размере дельты и текущей рабочей нагрузке, система решает, когда выполнять слияние основной части данных столбца с ее дельтой. Во время выполнения этой операции используется новая дельта.

SAP рекомендует использовать хранилище по строкам для небольших таблиц, в которых строки часто обновляются или удаляются. Таблицы, которые хранятся по строкам, всегда сохраняются в основной памяти. Однако таблицы, которые хранятся по столбцам, могут быть очень большими, и они могут быть разбиты на два раздела, один из которых хранится в памяти, а другой – на дисках. SAP называет раздел таблицы в основной памяти текущим (current), а на диске – историческим (historical) (другие поставщики используют термины горячий (hot) и холодный (cold) соответственно). Имеется в виду, что текущая часть таблицы должна содержать данные, которые в настоящее время активно используются. HANA может распределять данные между этими разделами автоматически в зависимости от времени жизни определенных строк таблицы или с использованием явных подсказок, предоставляемых приложениями. Текущие и исторические разделы имеют разные схемы хранения, но могут быть доступны с помощью одного запроса.

Имеется несколько дополнительных архитектурных решений, позволяющих одновременно и эффективно обрабатывать транзакции и аналитические запросы в одной системе. Во-первых, каждый аналитический запрос компилируется и оптимизируется с учетом самой последней статистики базы данных. Запросы, используемые в повторяющихся (параметризованных) транзакциях, подготавливаются заранее; при обработке транзакции используются готовые исполняемые планы запросов.

Во-вторых, аналитические запросы обрабатываются с высокой степенью распараллеливания с использованием нескольких потоков, выполняемых на разных ядрах ЦП. Распараллеливание транзакционных запросов считается вредным: переключение контекста обходится слишком дорого. Поэтому каждый транзакционный (простой) запрос полностью выполняется в одном потоке.

Наконец, чтобы минимизировать негативные эффекты конкуренции между операциями транзакций и аналитическими запросами, которые могут замедлить обработку транзакций, транзакционные операции всегда имеют более высокий приоритет, чем запросы OLAP.

4.1.3 HANA и энергонезависимая основная память

SAP HANA – первая коммерческая СУБД, активно использующая NVM (Intel Optane PMem) [29-30]. Учитывая особенности Optane NVM (высокая задержка и большая емкость), разработчики HANA решили использовать эту память для замены основных частей поколоночных хранилищ. Основными преимуществами этого подхода являются очень быстрый перезапуск системы после отключения питания (энергонезависимость) и возможность расширения текущих разделов столбчатых таблиц (большая емкость). Минус – очень вероятное замедление аналитики (большая задержка).

Авторы [29] отмечают, что замена DRAM на NVM может:

- сильно упростить и ускорить поддержку долговечности транзакций;
- существенно увеличить объем памяти на один процессор по разумной цене;
- предоставлять доступ к данным непосредственно в NVRAM после перезапуска, без необходимости перезагружать их с диска в DRAM.

Вторая и третья возможности уже реализованы с использованием существующего в настоящее время Optane PMem. Для реализации первой (очень важной) возможности нужны новые версии NVM с лучшими характеристиками задержки.

4.2 Аналитика реального времени в основной памяти в Microsoft

Похоже, что Microsoft внедрила аналитику в реальном времени в SQL Server в три этапа.

Во-первых, они разработали поколоночное хранилище таблиц на дисках [31]. Затем был разработан и реализован механизм OLTP в основной памяти Hekaton [32]. И, наконец, они разработали аналитический механизм для своего хранилища таблиц по столбцам и интегрировали все эти компоненты в SQL Server 2014 (и улучшили в SQL Server 2016), а также предоставили аналитику в реальном времени [33].

Судя по публикациям, Пер-Оке Ларсон (Per-Åke Larson) был руководителем проектов как хранилищ данных в основной памяти, так и систем для аналитики в реальном времени. Он ушел из Microsoft в 2015 году.

4.2.1 Архитектурные решения

Упрощенная архитектура SQL Server 2014 изображена на рис. 3.

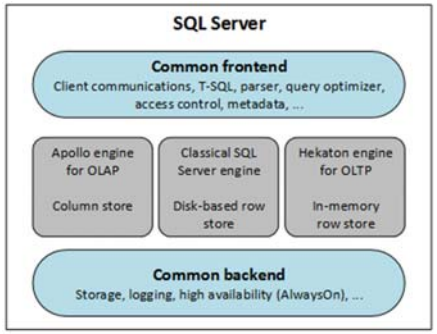


Рис. 3. Упрощенная архитектура SQL Server [33]
Fig. 3. A simplified architecture of SQL Server [33]

Аналитика в реальном времени обеспечивается компонентами Hekaton и Apollo. Hekaton – это специализированный механизм транзакционной базы данных в памяти, использующий хранилище строк. Apollo – это специализированная аналитическая дисковая подсистема, использующая хранилище по столбцам. Собственно, Apollo – это не название какого-либо продукта, это всего лишь кодовое название проекта. Рабочие данные очень быстро передаются из Hekaton в Apollo и преобразуются из представления на основе строк в представление на основе столбцов. (Последнюю операцию можно рассматривать как своего рода облегченный ETL.)

Основные архитектурные идеи Hekaton заключаются в следующем. Индексы (хэш и своего рода B-Tree) разработаны и оптимизированы для работы с данными, находящимися в основной памяти. Операции с индексами не журнализуются, и все индексы перестраиваются во время восстановления базы данных. Все внутренние структуры данных – распределители памяти, хеш-индексы и диапазоны, а также карта транзакций – полностью свободны от блокировок. Операторы SQL и хранимые процедуры компилируются в настраиваемый высокоэффективный машинный код.

Система определяет *горячие* и *холодные* данные и сохраняет в памяти только горячие данные [34]. Основная идея состоит в том, что не все данные транзакционной базы данных должны быть доступны одинаково быстро; некоторые из них могут быть помещены во внешнее хранилище без замедления обработки транзакций. Hekaton использует LRU-подобный алгоритм для определения того, какие данные не используются активно транзакциями, и перемещает соответствующие части таблиц в дисковое хранилище. Конечно, в памяти и на дисках используются разные физические схемы расположения строк. Поколоночное представление для аналитических целей поддерживается для обеих частей таблиц.

4.2.2 SQL-сервер и энергонезависимая основная память

Первые шаги по использованию NVM были сделаны до выпуска SQL Server 2014 для Linux. В то время Microsoft использовала небольшой объем NVM с прямым доступом, чтобы поместить в NVM буферизованный конец журнала, который еще не сброшен во внешнее хранилище [35]. Потеря хвоста журнала из-за отключения питания – хорошо известная проблема дисковых СУБД. Эта проблема вынуждает использовать очень сложные алгоритмы восстановления базы данных. Использование NVM гарантирует, что весь журнал будет доступен после отключения питания. Это радикально упрощает как поддержку долговечности транзакций, так и процесс восстановления базы данных.

Позже поддержка NVM в SQL Server была значительно расширена. SQL Server 2019 для Linux [36] предоставляет возможность разместить любую базу данных или файл журнала в NVM. Что еще более важно, SQL Server теперь поддерживает так называемый гибридный буферный пул, в котором все очищенные страницы (уже сброшенные на внешнее хранилище) перемещаются в NVM. После этого эти страницы остаются доступными без выполнения операций ввода-вывода.

4.3 Oracle database in memory

Решение Oracle для управления данными в основной памяти основано на in-memory СУБД TimesTen [37]. Проект TimesTen был основан в Hewlett-Packard Labs Мари-Анн Неймат (Marie-Anne Neimat). В то время система называлась SmallDB [38] и предназначалась для использования во внутренних целях HP. В 1996 году Неймат основала стартап TimesTen Inc. С этого времени TimesTen активно использовалась в качестве транзакционной СУБД реального времени во многих приложениях, особенно в области телекоммуникаций.

В 2005 году TimesTen была приобретена Oracle, интегрирована с общей инфраструктурой Oracle и начала использоваться в качестве кеш-памяти основной СУБД Oracle. Около десяти лет Неймат продолжала руководить проектом TimeTen, но затем ушла из Oracle.

Интегрированная опция Oracle Database In-Memory, поддерживающая функциональность HTAP, впервые появилась в Oracle 12c в 2013 году [39]. В настоящее время Тиртханкар Лахири (Tirthankar Lahiri) и Шасанк Чаван (Shasank Chavan) (оба являются вице-президентами Oracle), как кажется, руководят всем направлением Oracle in-memory.

4.3.1 Архитектурные особенности

Система поддерживает два вида хранилищ данных: хранилище строк находится на дисках, а хранилище столбцов – в основной памяти. Можно выборочно объявить важные таблицы или их разделы резидентными в хранилище столбцов при использовании буферного кеша для остальной части таблицы. Таким образом, есть имеются две области основной памяти для размещения частей базы данных: традиционный кеш для блоков внешней памяти и хранилище столбцов.

Части одной и той же таблицы, хранящиеся в разных представлениях (и в разных хранилищах), обновляются синхронно. Хранилище строк обеспечивает быстрый транзакционный доступ к данным с помощью тщательно разработанных индексов и кэширования блоков в памяти. Хранилище столбцов оптимизировано для обработки аналитических запросов. Свежесть данных для аналитики обеспечивается автоматической синхронизацией содержимого хранилищ столбцов и строк при вставке, удалении или обновлении строк. Многоверсионность, используемая в управлении транзакциями, позволяет снизить конкуренцию за данные между транзакциями и аналитическими запросами.

Oracle Database In-Memory может быть развернута в инфраструктуре Oracle Real Application Cluster (RAC) [40]. Общая архитектура представлена на рис. 4.

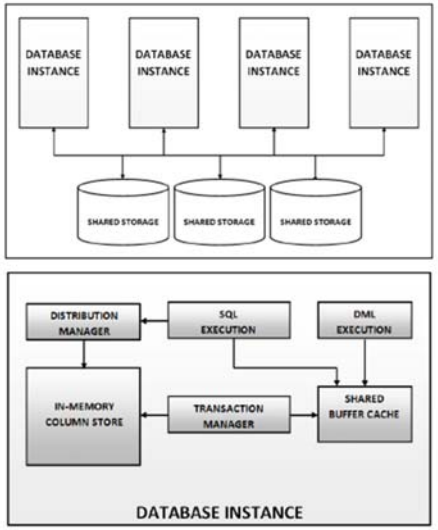


Рис. 4. Архитектура Database In-Memory on Oracle RAC [40]
Fig. 4. Architecture of Database In-Memory on Oracle RAC [40]

В этом случае блоки данных, содержащие части хранилища строк, доступны и изменяются через общий буферный кеш базы данных. Кроме того, для каждого отдельного экземпляра базы данных можно образовать индивидуальное хранилище столбцов в основной памяти. Все таблицы горизонтально разделяются по всем узлам кластера.

Если система работает в режиме in-memory, Exadata Database Machine автоматически переформатирует все данные, к которым осуществляется доступ, в поколоночный формат в основной памяти и сохраняет их в так называемом флэш-кеше (внешнее хранилище большого объема на основе флэш-памяти) [41]. После этого все части запросов, выгружаемые в Exadata, выполняются так же, как если бы они обрабатывались в основной памяти на узле RAC. В частности, для обработки столбцов таблицы Exadata использует инструкции SIMD.

4.3.2 Oracle и энергонезависимая основная память

Oracle предоставляет опцию Persistent Memory Database в своей базе данных Database 20c [42]. Фактически, этот вариант дает возможность разместить всю базу данных или ее часть в NVM. Точнее, они предоставляют средство PMEM Filestore, которое на самом деле обеспечивает файловую систему в энергонезависимой основной памяти, поддерживающую атомарные обновления блоков данных базы данных.

PMEM Filestore предоставляет внешний интерфейс для доступа к базе данных Oracle непосредственно в постоянной памяти. СУБД выполняет ввод-вывод в хранилище PMEM путем копирования памяти намного быстрее, чем ввод-вывод через традиционные вызовы операционной системы. Конечно, доступ к PMEM Filestore предоставляется без традиционного блочного кэширования.

4.4 DB2 с ускорителем BLU

В IBM DB2 ускорение аналитики осуществляется с помощью хранилища столбцов (в памяти), многоядерного распараллеливания и аппаратной векторной обработки. Работа над собственной поколоночной подсистемой в основной памяти в IBM началась в 2006 году. До

2016 года менеджером и техническим руководителем этого направления был Гай Ломан (Guy Lohman) (затем он ушел на пенсию).

Первым продуктом в памяти, разработанным командой Ломана, был Blink [43]. Это чистый механизм хранения столбцов в основной памяти. Он успешно используется и сейчас, например, как Informix Warehouse Accelerator [44].

BLU [45] – это второе поколение продукта. Общая архитектура DB2 с BLU показана на рис. 5 [46].

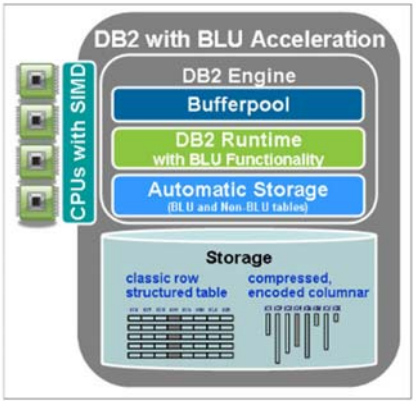


Рис. 5. Архитектура DB2 с BLU [46]
Fig. 5. The architecture of DB2 with BLU [46]

В базе данных, управляемой DB2 с помощью BLU, каждая таблица может храниться либо в традиционном хранилище строк, либо в хранилище столбцов BLU. Важной особенностью BLU является то, что таблицы в хранилище столбцов могут быть больше, чем размер доступной основной памяти. Фактически, все таблицы хранятся в блоках внешней памяти и доступны через обычный блочный кэш. Однако BLU пытается полностью обработать каждый запрос в основной памяти (хотя технически в этом нет необходимости) и обеспечивает соответствующее управление рабочей нагрузкой. Чтобы обеспечить доступность всех необходимых данных в памяти, BLU использует агрессивную политику предварительной выборки.

Другие характеристики DB2 с BLU кажутся традиционными для современных in-memory СУБД, хотя разработчики BLU подчеркивают очень высокую эффективность их реализации. Система поддерживает возможность сканирования и сравнения сжатых данных. Разработчики избегают блокировок, чтобы обеспечить максимальное распараллеливание. Все структуры данных оптимизированы для минимизации промахов в кэше данных и инструкций. Активно используются векторные команды.

Мы не смогли найти в публикациях и документации по DB2 никаких упоминаний об использовании NVM (или, по крайней мере, о планах использования). Единственное, что касается NVM, – это возможное использование SSD на базе Optane.

4.5 Altibase

Altibase Corporation – южнокорейская компания, основанная в 1999 году. Их in-memory СУБД была приобретена у Южнокорейского научно-исследовательского института электроники и телекоммуникаций (South Korean Electronics and Telecommunications Research Institute). С 2005 года компания предлагает гибридную in-memory и on-disk СУБД Altibase. С 2018 года Altibase является продуктом с открытым исходным кодом [47].

Altibase – это гибридная СУБД в том смысле, что она может управлять данными, хранящимися в памяти, или данными, хранящимися на дисках, или обоими видами данных одновременно. В обоих случаях система хранит таблицы только по строкам [48].

Систему можно рассматривать как разновидность СУБД HTAP, потому что:

- она обеспечивает очень высокую скорость обработки транзакций при работе в режиме основной памяти, и
- дает возможность анализировать свежие операционные данные (в основной памяти) вместе с историческими данными (на дисках) при работе в смешанном режиме.

Однако Altibase, безусловно, не является аналитической системой в реальном времени, поскольку она не имеет хранилища столбцов в основной памяти и, следовательно, не может обеспечивать быструю аналитику. Кроме того, система, вероятно, использует традиционный механизм блокировки, поскольку в документации упоминается обнаружение синхронизационных тупиков.

4.6 MemSQL (SingleStore)

СУБД MemSQL [49], переименованная в SingleStore осенью 2020 г., разработана стартапом MemSQL Software, основанным в 2011 году Эриком Френкилем (Eric Frenkiel) и Никитой Шамгуновым (Nikita Shamgunov) (ранее они оба работали в Facebook). Первый выпуск системы был анонсирован в 2013 году. MemSQL совместима со стандартным протоколом соединения баз данных MySQL (MySQL Wire Protocol). Это означает, что большинство драйверов и продуктов MySQL и MariaDB работают с MemSQL.

На рис. 6 изображена архитектура MemSQL, которая кажется одновременно очень привлекательной для разработчиков современных приложений и немного перегруженной. Здесь нас интересует только основная часть этой архитектуры.

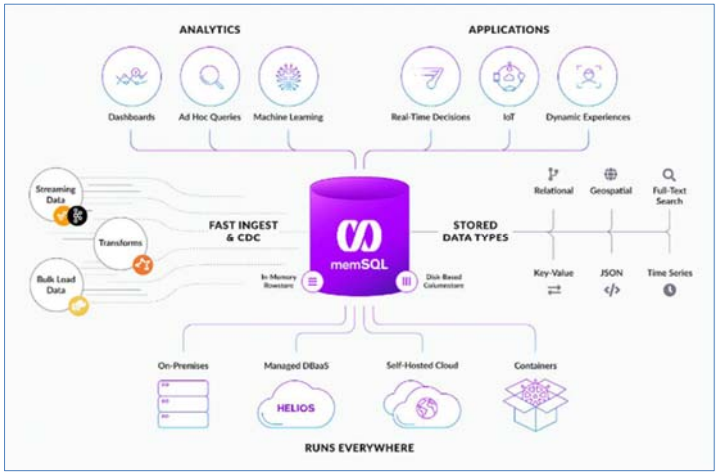


Рис. 6. Архитектура SingleStore [49]
Fig. 6. SingleStore architecture [49]

Система поддерживает хранение таблиц по строкам в основной памяти и по столбцам на дисках. Хранилище строк используется для обработки транзакций. Аналитические запросы могут дополнительно ссылаться на поколоночные таблицы, хранящиеся на дисках (они могут содержать, например, исторические данные). Разработчики заявляют, что их система является HTAP и аналитической в реальном времени. Да, очевидно, это СУБД HTAP. Однако

мы не уверены, что система действительно может предоставлять аналитику в реальном времени из-за ее основной ориентации на обработку транзакций (хранилище строк в памяти).

Также стоит отметить, что MemSQL позиционируется как распределенная система без совместного использования ресурсов. На наш взгляд, это хорошо для масштабной аналитики, поддерживаемой традиционным способом. Однако в такой архитектуре сложно одновременно поддерживать очень быструю обработку транзакций и очень быструю обработку аналитических запросов. В документации MemSQL ничего не говорится о распределенных транзакциях, двухфазных фиксациях и т.д. А утверждения [50] об очень быстрых распределенных соединениях и агрегации не кажутся убедительными (все это может быть очень быстрым для распределенной СУБД, но слишком медленным для анализа данных в реальном времени).

Разработчики MemSQL думают об использовании постоянной памяти, но не собираются никаким образом переписывать свой код; они предлагают использовать Optane для увеличения емкости основной памяти [51].

4.7 HyPer

Мы заканчиваем этот раздел рассмотрением трех академических HTAP-СУБД, одна из которых, HyPer, является достаточно зрелой системой, а две другие – только исследовательскими прототипами, но они очень интересны. Начнем с краткого обсуждения HyPer.

Проект HyPer реализуется командой баз данных Мюнхенского технического университета под руководством Альфонса Кемпера (Alfons Kemper) и Томаса Ноймана (Thomas Neumann). Хотя это университетский проект, он не является открытым. Исходные тексты системы никогда не публиковались. Более того, в 2015 г. был создан околоуниверситетский стартап HyPer, а в 2016 г. он был поглощен компанией Tableau Software, которая свой вариант СУБД называет Hyper. Детали этой сделки нам неизвестны, но так или иначе развитие HyPer в Мюнхенском университете продолжается. Первый известный отчет по проекту [52] был опубликован в 2010 году.

HyPer – настоящая in-memory СУБД для многоядерных компьютеров. Система поддерживает хранилища таблиц как по строкам, так и по столбцам. Исходный формат таблицы может быть выбран при создании таблицы, и физическая структура этой таблицы может меняться в зависимости от рабочей нагрузки.

Первая версия архитектуры HyPer была очень простой и элегантной, хотя и имела некоторые ограничения. Основные идеи заключались в следующем [53].

Предполагалось, что большинство транзакций могут быть подготовлены заранее, содержать только несколько простых запросов и производить доступ только к нескольким кортежам. Такие транзакции называются короткими. Все короткие транзакции выполняются последовательно в основном процессе OLTP. Вся область основной памяти, в которой находится база данных, отображается в виртуальную памяти этого процесса.

Процесс OLTP периодически (раз в несколько секунд) порождает процесс OLAP после фиксации некоторой только что завершенной транзакции. После этого процесс OLTP включает механизм копирования при записи (copy-on-write), который предоставляет новую страницу основной памяти при любой первой операции записи в соответствующую страницу виртуальной памяти. Таким образом, вновь созданный процесс OLAP видит в своей виртуальной памяти согласованный образ базы данных (согласованный моментальный снимок, consistent snapshot). Процесс OLAP выполняет аналитические запросы над этим моментальным снимком, свежесть данных которого соответствует периодичности создания новых процессов OLAP.

Транзакция, которая выполняет слишком много запросов или обращается к слишком большому количеству кортежей, считается длинной. Выполняемая длинная транзакция

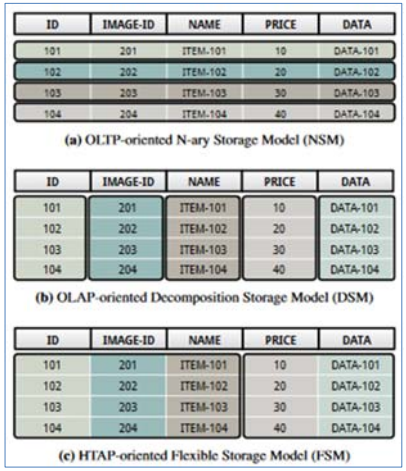
откатывается и перенаправляется в текущий процесс OLAP, который имитирует ее выполнение на основе текущего согласованного моментального снимка. Затем эта (частично измененная) транзакция снова перенаправляется в общую очередь транзакций как короткая транзакция. Когда она выполняется, система сначала проверяет все смоделированные обновления по фактическому состоянию базы данных, а затем, если эта проверка прошла успешно, выполняет все эти обновления.

Аналитические запросы распараллеливаются по всем доступным ядрам с использованием массивно-параллельной версии хорошо известного алгоритма сортировки со слиянием (sort-merge join) [54].

Позже разработчики HyPer изменили эту простую архитектуру, чтобы обеспечить более высокую пропускную способность для транзакционных рабочих нагрузок. В частности, они включили в систему свою собственную версию версионного алгоритма управления параллелизмом [55], которая значительно усложняет систему, но улучшает ее транзакционные характеристики.

4.8 Peloton

Проект выполнялся группой баз данных университета Карнеги-Меллон под руководством Эндрю Павло (Andrew Pavlo). Peloton – это in-memory HTAP-СУБД, которая поддерживает хранилища строк и столбцов (см. рис. 7) и обеспечивает самоуправление всех компонентов системы на основе машинного обучения и других технологий искусственного интеллекта [56].



ID	IMAGE-ID	NAME	PRICE	DATA
101	201	ITEM-101	10	DATA-101
102	202	ITEM-102	20	DATA-102
103	203	ITEM-103	30	DATA-103
104	204	ITEM-104	40	DATA-104

(a) OLTP-oriented N-ary Storage Model (NSM)

ID	IMAGE-ID	NAME	PRICE	DATA
101	201	ITEM-101	10	DATA-101
102	202	ITEM-102	20	DATA-102
103	203	ITEM-103	30	DATA-103
104	204	ITEM-104	40	DATA-104

(b) OLAP-oriented Decomposition Storage Model (DSM)

ID	IMAGE-ID	NAME	PRICE	DATA
101	201	ITEM-101	10	DATA-101
102	202	ITEM-102	20	DATA-102
103	203	ITEM-103	30	DATA-103
104	204	ITEM-104	40	DATA-104

(c) HTAP-oriented Flexible Storage Model (FSM)

Рис. 7. Различные схемы хранения таблиц в Peloton [57]
Fig. 7. Different table storage layouts in Peloton [57]

Еще одна важная особенность Peloton – это практически полноценная поддержка NVM вместо (или, вернее, вместе с) энергозависимой основной памяти (или, вернее, вместе с ней). Эту часть проекта реализовал Джой Арулрадж (Joy Arulraj) (совместно с Павло) [58]. За эту работу Арулрадж получил в 2019 году премию Джима Грея за лучшую докторскую диссертацию (Jim Gray Doctoral Dissertation Award).

Наиболее интересными представляются следующие аспекты этой работы:

- архитектура подсистемы хранения использует свойства долговечности и байтовой адресации NVM; обеспечивается экономичное использование энергонезависимой памяти и увеличение срока ее службы за счет уменьшения количества операций записи;

- в системе используется новый протокол журнализации и восстановления, который называется «журнализация с отложенной записью» (Write-Behind Logging, WBL), что позволяет достичь высокого уровня доступности.

Последняя версия Peloton доступна для загрузки из общедоступного репозитория Github [59]. Однако около двух лет назад команда решила отказаться от развития этого репозитория и перейти к созданию новой СУБД. Новый репозиторий создан [60], и ведется работа над проектом под названием NoisePage.

4.8 SOFORT

Проект SOFORT гибридной системы NVM-DRAM с хранением таблиц по стлбцам [61] был выполнен Исмаилом Укидом (Ismail Oukid) в группе баз данных Дрезденского технического университета под руководством Вольфганга Ленера.

В целом SOFORT представляет собой систему с поколоночным хранением таблиц категории HTAP (рис. 8). Система поддерживает одноуровневую подсистему хранения, использующую как NVM, так и традиционную DRAM. SOFORT – это незавершенный исследовательский проект (он мертв).

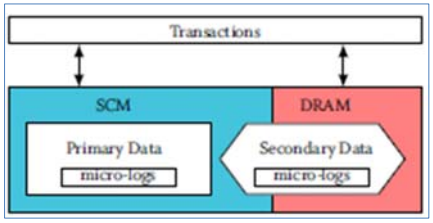


Рис. 8. Гибридная архитектура NVM-DRAM SOFORT [61]
Fig. 8. Hybrid NVM-DRAM architecture of SOFORT [61]

Тем не менее, Укид сделал несколько интересных вещей, которые, безусловно, будут полезны при разработке будущих in-NVM СУБД:

- модель программирования с использованием NVM;
- управление энергонезависимой памятью и ее распределение для нужд СУБД;
- управление транзакциями и восстановление базы данных после сбоев;
- фреймворк для тестирования программного обеспечения, ориентированного на использование энергонезависимой памяти.

5. Обсуждение и заключение

Как видно из приведенного выше обзора, существует множество способов реализации функций HTAP и предоставления аналитики в реальном времени. Общие принципы заключаются в следующем:

- предоставлять свежие данные для анализа путем объединения транзакционной и аналитической обработки в одной системе баз данных;
- сделать обработку транзакций максимально быстрой, чтобы удовлетворить потребности транзакционных клиентов и повысить актуальность данных;
- сделать оперативные данные доступными для анализа как можно быстрее, чтобы удовлетворить первое требование аналитики в реальном времени – свежесть данных;
- как можно быстрее производить оценку аналитических запросов для удовлетворения второго требования аналитики в реальном времени – быстрого анализа данных.

Требования к HTAP-СУБД и аналитике в реальном времени не включают никаких абсолютных чисел, но имеют некоторые неформализованные разумные количественные ограничения. Мы не знаем точно, сколько транзакций (и какого типа) в секунду должна обрабатывать HTAP-СУБД и каково максимально допустимое время для выполнения аналитического запроса. Вот почему этим (полу) качественным требованиям может удовлетворять множество различных архитектур: все они делают все возможное в рамках этих неформальных границ.

Обычный подход к удовлетворению этих требований, насколько возможно их удовлетворить, состоит в том, чтобы хранить всю базу данных в основной памяти. Этот выбор позволяет почти избежать любого ввода-вывода с внешними устройствами хранения (постоянное хранилище используется только для обеспечения долговечности транзакций) и использовать прямые указатели на объекты базы данных в базе данных. Все внутренние структуры данных должны разрабатываться с учетом кеширования.

Для дальнейшего повышения производительности используются передовые методы оптимизации запросов, своевременная компиляция запросов в машинный код, многоядерное распараллеливание, инструкции SIMD и аппаратные ускорители, такие как графические процессоры и FPGA.

Обычно одна СУБД поддерживает два хранилища данных – строковое и столбцовое. Этот подход представляет собой компромисс между возможностью быстрой обработки транзакций (хранилище строк) и быстрой аналитикой запросов (хранилище столбцов), с одной стороны, и необходимостью преобразования данных из строкового формата в поколоночный до того, как данные станут доступны для аналитики, с другой стороны.

Чтобы избежать высокого уровня конкуренции между одновременно выполняемыми транзакциями, а также между транзакциями и параллельно выполняемыми аналитическими запросами, обычно используется какой-то вид многоверсионного управления параллелизмом, часто оптимистический. Все внутренние объекты базы данных, такие как индексы, распределители памяти и т.д., проектируются так, чтобы избежать всех видов блокировок.

Однако хранение базы данных полностью в памяти также является компромиссом между высокой производительностью СУБД и ограниченным размером базы данных. Чтобы смягчить это ограничение, основные производители предпочитают комбинировать хранилища в оперативной памяти и на диске. Они утверждают, что после этого производительность СУБД не ухудшается (непонятно почему), но в любом случае архитектура системы становится намного сложнее.

Есть несколько попыток предоставить функции HTAP и аналитики в реальном времени в массивно-параллельных архитектурах без совместного использования ресурсов. Мы считаем, что такая цель труднодостижима. Во-первых, даже если каждый узел такой системы полностью хранит базу данных в памяти, неизбежное сетевое взаимодействие значительно снизит производительность системы. Во-вторых, как транзакционные, так и аналитические СУБД без совместного использования ресурсов полагаются на подходящее разделение данных по узлам системы. Однако цели разделения для транзакционных и аналитических СУБД различны. Транзакционная СУБД пытается разделить базу данных, чтобы минимизировать количество распределенных транзакций. Аналитическая СУБД при разделении базы данных пытается минимизировать количество распределенных объединений. Сомнительно, чтобы в одной системе можно было достичь обеих целей.

Наконец, значительное число поставщиков СУБД HTAP в оперативной памяти уже используют в своих продуктах доступную в настоящее время энергонезависимую основную память или планируют использовать ее в будущем. Наш обзор демонстрирует достаточно широкий спектр вариантов использования NVM от простого расширения памяти с помощью NVM до наиболее многообещающих одноуровневых архитектур хранения. Однако

последний вариант использования сейчас (частично) реализован только в исследовательских проектах.

Причина, вероятно, в том, что в настоящее время на рынке доступен только один вид NVM – Intel Optane, основанный на технологии PCM. Эти модули DIMM на основе NVM имеют довольно высокую задержку и не могут заменить DRAM. Мы полагаем, что широко ожидаемое новое поколение NVM (возможно, на основе STT) значительно ускорит внедрение NVM в HTAP (и чисто транзакционных) СУБД.

Данная статья является русскоязычным (авторским) вариантом ранее опубликованной статьи [62], в которой, кроме смены языка, исправлены некоторые незначительные неточности.

Список литературы / References

[1] Michael Stonebraker, Ugur Cetintemel. "One Size Fits All": An Idea Whose Time Has Come and Gone. Proceedings of the 21st International Conference on Data Engineering, 2005, pp. 2-11.

[2] Andrew Lamb, Matt Fuller, Ramakrishna Varadarajan, Nga Tran, Ben Vandiver, Lyric Doshi, Chuck Bea. The Vertica Analytic Database: C-Store 7 Years Later. Proceedings of the VLDB Endowment, vol. 5, no. 12, 2012, pp. 1790-1801.

[3] Michael Stonebraker, Ariel Weisberg. The VoltDB Main Memory DBMS. Bulletin of the Technical Committee on Data Engineering, vol. 36, no. 2, 2013, pp. 21-27.

[4] Franz Faerber, Alfons Kemper, Per-Åke Larson, Justin Levandoski, Thomas Neumann, Andrew Pavlo. Main Memory Database Systems. Foundations and Trends in Databases, vol. 8, no. 1-2, 2016, pp. 1–130.

[5] С.Д. Кузнецов. В ожидании нативных архитектур СУБД на основе энергонезависимой основной памяти. Труды ИСП РАН, том 32, выпуск 1, 2020 г., стр. 153-180. DOI: 10.15514/ISPRAS-2020-32(1)-9 / Sergey Kuznetsov. Towards a Native Architecture of in-NVM DBMS. Proceedings of the 6th International Conference on Actual Problems of Systems and Software Engineering (APSSE), 2019, pp. 77-89.

[6] Gartner Glossary: Real-time Analytics. URL: https://www.gartner.com/en/information-technology/glossary/real-time-analytics, accessed 08-16-2020.

[7] Mohammed Al-Kateb, Paul Sinclair, Grace Kwan-On Au, Carrie Ballinger. Hybrid Row-Column Partitioning in Teradata. Proceedings of the VLDB Endowment, vol. 9, no. 13, 2016, pp. 1353-1364.

[8] Hybrid transactional/analytical processing. From Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Hybrid_transactional/analytical_processing, accessed 08-17-2020.

[9] Gartner Glossary: HTAP-enabling In-memory Computing Technologies. URL: https://www.gartner.com/en/information-technology/glossary/htap-enabling-memory-computing-technologies, accessed 08-17-2020.

[10] Jan Lindström, Vilho Raatikka, Jarmo Ruuth, Petri Soini, 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, 2013, pp. 14-20.

[11] Stephen Tu, Wenting Zheng, Eddie Kohler, Barbara Liskov, and Samuel Madden. Speedy Transactions in Multicore In-Memory Databases. Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, 2013, pp. 18-32.

[12] EXASOL: A Peek Under the Hood. White Paper. URL: https://www.dataviz.sk/wp-content/uploads/2019/09/WP_Exasol_Technical_Peek_under_the_hood.pdf, accessed 08-17-2020.

[13] The Official History of TM1. URL: https://cubewise.com/history/, accessed 08-17-2020.

[14] Michael Schrader, Dan Vlamis, Mike Nader, Chris Claterbos, Dave Collins, Mitch Campbell, Floyd Conrad. Oracle Essbase & Oracle OLAP: The Guide to Oracle's Multidimensional Solution. McGraw-Hill Education, 2009, 524 p.

[15] Yuan Zhou, Haodong Tang, Jian Zhang. Spark-PMoF: Accelerating big data analytics with Persistent Memory over Fabric. Strata Data Conference, 2019 .

[16] Hasso Plattner. A common database approach for OLTP and OLAP using an in-memory column database. Proceedings of the ACM SIGMOD International Conference on Management of data, 2009, pp. 1–2.

[17] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, Pat Helland. The End of an Architectural Era (It's Time for a Complete Rewrite). Proceedings of VLDB, 2007, pp. 1150–1160.

[18] solidDB in a Nutshell. URL: https://www.teamblue.unicomsi.com/index.php/download_file/499/660/, accessed 08-19-2020.

[19] gunaprsd/silo: Multicore in-memory storage engine. URL: https://github.com/stephentu/silo, accessed 08-19-2020.

[20] Oracle Exalytics In-Memory Machine: A Brief Introduction. Oracle White Paper, 2013. URL: https://www.oracle.com/technetwork/middleware/bi/overview/whitepaper-exalytics-x3-4-1973011.pdf, accessed 08-19-2020.

[21] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. Spark: Cluster Computing with Working Sets. Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, 2010, pp. 1-7.

[22] Shuffle Remote PMem Extension for Apache Spark Guide. URL: https://github.com/Intel-bigdata/OAP/tree/master/oap-shuffle/RPMem-shuffle, accessed 08-22-2020.

[23] Vishal Sikka. Timeless Software. URL: http://vishalsikka.blogspot.com/2008/10/timeless-software.html, accessed 08-23-2020.

[24] Frederik Transier, Peter Sanders. Engineering basic algorithms of an in-memory text search engine. ACM Transactions on Information Systems, 2010, Article No. 2.

[25] J. Andrew Ross. SAP NetWeaver BI Accelerator. SAP PRESS, 2008, 260 p.

[26] Sang K. Cha and Changbin Song. P*TIME: Highly Scalable OLTP DBMS for Managing Update-Intensive Stream Workload. Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004, pp. 1033-1044.

[27] André Bögelsack, Stephan Gradl, Manuel Mayer, Helmut Krcmar. SAP MaxDB Administration. SAP PRESS, 2009, 326 p.

[28] Franz Faerber, Norman May, Wolfgang Lehner, Philipp Große, Ingo Müller, Hannes Rauhe, Jonathan Dees. The SAP HANA Database – An Architecture Overview. Bulletin of the Technical Committee on Data Engineering, March 2012, vol. 35, no. 1, pp. 28-33.

[29] Mihnea Andrei, Christian Lemke, Günter Radestock, Robert Schulze, Carsten Thiel, Rolando Blanco, Akanksha Meghlan, Muhammad Sharique, Sebastian Seifert, Surendra Vishnoi, Daniel Booss, Thomas Peh, Ivan Schreter, Werner Thesing, Mehul Wagle, Thomas Willhalm. SAP HANA Adoption of Non-Volatile Memory. Proceedings of the VLDB Endowment, vol. 10, no. 12, 2017, pp. 1754-1765.

[30] Intel Optane Persistent Memory and SAP HANA Platform Configuration. Configuration Guide. 2019. URL: https://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/sap-hana-and-intel-optane-configuration-guide.pdf, accessed 08-26-2020.

[31] Per-Åke Larson, Cipri Clinciu, Eric N. Hanson, Artem Oks, Susan L. Price, Srikumar Rangarajan, Aleksandras Surna, Qingqing Zhou. SQL Server Column Store Indexes. Proceedings of the ACM SIGMOD International Conference on Management of data, 2011, pp. 1177-1184.

[32] Per-Åke Larson, Mike Zwilling, Kevin Farlee. The Hekaton Memory-Optimized OLTP Engine. Bulletin of the Technical Committee on Data Engineering, vol. 36, no. 2, 2013, pp. 34-40.

[33] Per-Åke Larson, Adrian Birka, Eric N. Hanson, Weiyun Huang, Michal Nowakiewicz, Vassilis Papadimos. Real-Time Analytical Processing with SQL Server. Proceedings of the VLDB Endowment, vol. 8, no. 12, 2015, pp. 1740-1751.

[34] Ahmed Eldawy, Justin Levandoski, Per-Åke Larson. Trekking Through Siberia: Managing Cold Data in a Memory-Optimized Database. Proceedings of the VLDB Endowment, vol. 7, no. 11, pp. 931-942.

[35] Bob Dorr. How It Works (It Just Runs Faster): Non-Volatile Memory SQL Server Tail of Log Caching on NVDIMM. URL: https://docs.microsoft.com/ru-ru/archive/blogs/bobsql/how-it-works-it-just-runs-faster-non-volatile-memory-sql-server-tail-of-log-caching-on-nvdimm, accessed 08-27-2020.

[36] Kellyn Gorman, Allan Hirt, Dave Noderer, Mitchell Pearson, James Rowland-Jones, Dustin Ryan, Arun Sirpal, Buck Woody. Introducing Microsoft SQL Server 2019: Reliability, scalability, and security both on premises and in the cloud. Packt Publishing, 2020, 488 p.

[37] Tirthankar Lahiri, Marie-Anne Neimat, Steve Folkman. Oracle TimesTen: An In-Memory Database for Enterprise Applications. Bulletin of the Technical Committee on Data Engineering, vol. 36, no. 2, 2013, pp. 6-13.

[38] Sherry Listgarten and Marie-Anne Neimat. Modelling Costs for a MM-DBMS. Proceedings of the International Workshop on Real-Time Databases, Issues and Applications (RTDB), 1996, pages 72-78.

[39] Tirthankar Lahiri, Shasank Chavan, Maria Colgan, Dinesh Das, Amit Ganesh, Mike Gleeson, Sanket Hase, Allison Holloway, Jesse Kamp, Teck-Hua Lee, Juan Loaiza1, Neil Macnaughton, Vineet Marwah, Niloy Mukherjee, Atrayee Mullick, Sujatha Muthulingam, Vivekanandhan Raja, Marty Roth, Ekrem Soylemez, Mohamed Zait. Oracle Database In-Memory: A dual format in-memory database. Proceedings of the IEEE 31st International Conference on Data Engineering, Seoul, 2015, pp. 1253-1258.

[40] Niloy Mukherjee, Shasank Chavan, Maria Colgan, Dinesh Das, Mike Gleeson, Sanket Hase, Allison Holloway, Hui Jin, Jesse Kamp, Kartik Kulkarni, Tirthankar Lahiri, Juan Loaiza, Neil Macnaughton, Vineet Marwah, Atrayee Mullick, Andy Witkowski, Jiaqi Yan, Mohamed Zait. Distributed Architecture of Oracle Database In-memory. Proceedings of the VLDB Endowment, vol. 8, no. 12, 2015, pp. 1630–1641.

[41] Shasank Chavan, Gurmeet Goindi. Oracle Database In-Memory on Exadata: A Potent Combination. Oracle OpenWorld 2018. URL: https://www.oracle.com/technetwork/database/exadata/pro4016-exadataandinmemory-5187037.pdf, accessed 08-28-2020.

[42] Oracle Database 20c. Database Administrator's Guide. Using Persistent Memory Database. URL: https://docs.oracle.com/en/database/oracle/oracle-database/20/admin/index.html, accessed 08-28-2020.

[43] Ronald Barber, Peter Bendel, Marco Czech, Oliver Draese, Frederick Ho, Namik Hrle, Stratos Idreos, Min-Soo Kim, Oliver Koeth, Jae-Gil Lee, Tianchao Tim Li, Guy Lohman, Konstantinos Morfonios, Rene Mueller, Keshava Murthy, Ippokratis Pandis, Lin Qiao, Vijayshankar Raman, Richard Sidle, Knut Stolze, Sandor Szabo. Business Analytics in (a) Blink. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 35, no. 1, 2012, pp. 9-14.

[44] IBM Informix Warehouse Accelerator. Technical white paper. URL: https://www.iiug.org/library/ids_12/IWA%20White%20Paper-2013-03-21.pdf, accessed 08-29-2020.

[45] Vijayshankar Raman, Gopi Attaluri, Ronald Barber, Naresh Chainani, David Kalmuk, Vincent KulandaiSamy, Jens Leenstra, Sam Lightstone, Shaorong Liu, Guy M. Lohman, Tim Malkemus, Rene Mueller, Ippokratis Pandis, Berni Schiefer, David Sharpe, Richard Sidle, Adam Storm, Liping Zhang. DB2 with BLU Acceleration: So Much More than Just a Column Store. Proceedings of the VLDB Endowment, Vol. 6, No. 11, 2013, pp. 1080-1091.

[46] Whei-Jen Chen, Brigitte Bläser. Marco Bonezzi, Polly Lau, Jean Cristie Pacanaro, Martin Schlegel, Ayesha Zaka, Alexander Zietlow. Architecting and Deploying DB2 with BLU Acceleration. IBM Redbooks, 2014, 420 p.

[47] Altibase. URL: https://github.com/ALTIBASE, accessed 08-30-2020.

[48] Altibase 7.1 Administrator's Manual. URL: https://github.com/ALTIBASE/Documents/blob/master/Manuals/Altibase_7.1/eng/Administrator's%20Manual%201.md, accessed 08-29-2020.

[49] MemSQL Software. The Cloud-Native Operational Database Built for Speed, Scale, and SQL. URL: https://www.memsql.com/resources/data_sheet-memsql_software/, accessed 08-30-2020.

[50] Jack Chen, Samir Jindel, Robert Walzer, Rajkumar Sen, Nika Jimsheleishvilli, Michael Andrews. The MemSQL Query Optimizer: A modern optimizer for real-time analytics in a distributed database. Proceedings of the VLDB Endowment, Vol. 9, No. 13, 2016, pp. 1401-1412.

[51] Eric Hanson. How to Use MemSQL with Intel's Optane Persistent Memory. URL: https://www.memsql.com/blog/how-to-use-memsql-with-intels-optane-persistent-memory/, accessed 08-30-2020.

[52] Alfons Kemper and Thomas Neumann. HyPer - Hybrid OLTP&OLAP High Performance Database System. Technical Report, TUM-I1010, Munich Technical University, 2010, 29 p.

[53] Alfons Kemper, Thomas Neumann, Jan Finis, Florian Funke, Viktor Leis, Henrik Mühe, Tobias Mühlbauer, Wolf Rödiger. Transaction Processing in the Hybrid OLTP&OLAP Main-Memory Database System HyPer. Bulletin of the Technical Committee on Data Engineering, vol. 36, no. 2, 2013, pp. 41-47.

[54] Martina-Cezara Albutiu, Alfons Kemper, Thomas Neumann. Massively Parallel Sort-Merge Joins in Main Memory Multi-Core Database Systems. Proceedings of the VLDB Endowment, vol. 5, no. 10, 2012, pp. 1064-1075.

[55] Thomas Neumann, Tobias Mühlbauer, Alfons Kemper. Fast Serializable Multi-Version Concurrency Control for Main-Memory Database Systems. Proceedings of the ACM SIGMOD International Conference on Management of data, 2015, pp. 677–689.

[56] 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 WuF, Ran Xian, Tieying Zhang. Self-Driving Database Management Systems. Proceedings of the 8th Biennial Conference on Innovative Data Systems Research, 2017, 6 p.

[57] Joy Arulraj. Andrew Pavlo. Prashanth Menon. Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads. Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 583–598.

[58] Joy Arulraj, Andrew Pavlo. Non-Volatile Memory Database Management Systems. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2019, 192 p.

[59] cmu-db / peloton. The Self-Driving Database Management System. URL: https://github.com/cmu-db/peloton, accessed 09-02-2020.

[60] cmu-db / terrier. URL: https://github.com/cmu-db/noisepage, accessed 05-06-2021.

[61] Ismail Oukid. Architectural Principles for Database Systems on Storage-Class Memory. Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn, 2019, pp. 477-486.

[62] S.D. Kuznetsov, P.E. Velikhov, and Q. Fu. Real-time analytics, hybrid transactional/analytical processing, in-memory data management, and non-volatile memory. In Proc. of the Ivannikov ISPRAS Open Conference, 2021, pp. 78-90. DOI: 10.1109/ISPRAS51486.2020.00019.

Информация об авторах / Information about authors

Сергей Дмитриевич КУЗНЕЦОВ – доктор технических наук, профессор, главный научный сотрудник ИСП РАН, профессор кафедр системного программирования МГУ, МФТИ и ВШЭ, старший научный сотрудник РЭУ им. Г.В. Плеханова. Научные интересы: управление данными, архитектуры систем управления данными, модели и языки данных, управление транзакциями, оптимизация запросов.

Sergey Dmitrievich KUZNETSOV – Doctor of Technical Sciences, Professor, Chief Researcher at ISP RAS, Professor at the Departments of System Programming of MSU, MIPT, and HSE, Senior Researcher at REU. Research interests: data management, architectures of data management systems, data models and languages, transaction management, query optimization.

Павел Евгеньевич ВЕЛИХОВ, ведущий инженер ключевых проектов. Научные интересы: массивно-параллельные СУБД, оптимизация запросов, методы искусственного интеллекта для оптимизации СУБД, полуструктурированные модели данных, in-memory СУБД.

Pavel Evgenievich VELIKHOV, Principal Engineer of Key Projects. Research interests: massively parallel DBMS, query optimization, artificial intelligence methods for DBMS optimization, semi-structured data models, in-memory DBMS.

Цян ФУ, представитель бизнес-подразделения. Научные интересы: массивно-параллельные СУБД.

Qiang FU, Business Representative. Research interests: massively parallel DBMS