АНАЛИЗ ВОЗМОЖНЫХ ПУТЕЙ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНЫХ СУБД КОНСЕРВАТИВНОГО ТИПА НА ПЛАТФОРМЕ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ

В.А. Райхлин, Р.Ш. Минязев, Р.К. Классен, А.В. Садовин

Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Россия

Обсуждаются вопросы повышения эффективности специализированных СУБД, ориентированных на обработку консервативных данных по результатам испытаний. Обсуждение проводится для двух возможных стратегий обработки запросов на платформе вычислительных кластеров с многоядерными узлами: 1) полное или частичное множество узлов кластера на один запрос и 2) один узел кластера на ограниченное по числу ядер множество запросов. За критерий эффективности принимается достижимая производительность и масштабируемость кластерной реализации. В рамках первой стратегии рассматривается последовательное улучшение характеристик СУБД *Clusterix*, разработанной в начале текущего века для платформы кластеров с одноядерными узлами, при переходе на платформу с *SMP*-узлами. Формулируются особенности асинхронной организации СУБД в разрабатываемой системе А'*Clusterix*. В рамках второй стратегии приводятся результаты, полученные для случая обработки запросов в узлах SUN-кластера под управлением *MySQL* при максимальной загрузке всех ядер.

Ключевые слова: параллельные СУБД, СУБД консервативного типа, *Clusterix*-прототип, распределенные БД, кластер с *SMP*-узлами, профилирование БД, асинхронное управление, *MySQL*-прототип.

Введение

Объемы хранимых данных по результатам разного рода испытаний (в том числе, – испытаний авиа-космической техники) могут быть весьма значительными. Их своевременная обработка требует применения высокопроизводительных вычислительных средств (HPC) и специализированного программного обеспечения – СУБД консервативного типа с эпизодическим обновлением данных. Дело в том, что обычно пользователям таких баз данных (БД) предоставляется только право чтения. Право записи в БД (ее изменения) является привилегированным. Аппаратные платформы (HPC-кластеры, хранилища данных) в настоящее время достаточно развиты. Вопросы же создания соответствующих СУБД все еще требуют серьезного обсуждения. Имеется множество разработок параллельных СУБД на платформе вычислительных кластеров (Microsoft SQL Server, IBM DB2 PDF, Oracle EXADATA, PostgreSQL Cluster, MySQL Cluster и др.). Но все они ориентированы, в основном, на выполнение множества сравнительно простых операций типа select и insert над динамически изменяемыми базами данных. Причина в том, что серьезной проблемой для высокопроизводительных реляционных СУБД всегда было и остается повышение скорости обработки сложных запросов [1, 2]. Для консервативных СУБД характерен высокий удельный вес именно сложных запросов типа «select – project – join», оперирующих множеством таблиц с большим числом операций соединения join. Определенное понимание необходимости развития работ в области СУБД консервативного типа уже имеется. Примером тому является проект специализированной системы SciDB для обработки научных данных (результатов испытаний, наблюдений за состоянием среды и др.) [3, 4].

Достаточно полно исследованным к настоящему времени представителем консервативных СУБД, реализующих стратегию «множество узлов кластера на один запрос», явля-

ется система Clusterix [5, 6]. Эта СУБД реализует регулярный план обработки запросов [7] с применением инструментальной СУБД MySQL на исполнительном уровне. Существенное повышение эффективности консервативной СУБД (рост масштабируемости по числу узлов, а потому и достижимой производительности многоузловой системы) достигается переходом к мультикластеризации с 2-процессорными узлами (один мощный узел на каждый запрос) [8]. Мощный узел реализуется как монокластер, работающий на грани масштабируемости. На одном достаточно мощном вычислительном кластере реализуется множество одновременно функционирующих СУБД Clusterix с репликацией между ними консервативной базы данных. Самостоятельное применение СУБД MySQL последних версий позволяет полностью использовать ресурсы многоядерных узлов с реализацией монокластера на одном узле (стратегия «один узел кластера на множество запросов» [9]). Ниже конкретизируются результаты, полученные авторами на платформе SAN-кластера. Обсуждаются перспективы продолжающихся исследований на той же платформе.

1. Множество узлов кластера на один запрос

За основу разработки СУБД Clusterix был принят проект [7, 10] многомашинной информационной системы, реализующей функции системы управления базами данных (машины баз данных). Система предназначена для параллельного выполнения запросов пользователя к базе данных по схеме «select – project – join». Обработка запросов – 2-уровневая. На нижнем уровне выполняются операции селекции (σ) и проекции (π) над исходными отношениями Ri базы данных. Результатом обработки нижнего уровня является промежуточное отношение Ri’. На верхнем уровне реализуется операция соединения

$$RBi–1 = Ri' \blacktriangleright\blacktriangleleft RBi–2 = \pi(\sigma\theta(Ri'xRBi–2)),$$

где RBi–1 и RBi–2 временные отношения как результаты соединений в i- и в (i-1)-шагах соответственно. Фильтрация на нижнем уровне значительно снижает объемы данных, передаваемых на верхний уровень. Отношения БД распределены по дискам на процессорах нижнего уровня (IOr). Распределение отношений осуществляется горизонтально с применением хеш-функции к первичному ключу для каждого кортежа отношения. В качестве такой хеш-функции используется функция деления по модулю [11]. Основание деления – число процессоров нижнего уровня (nIO). Остаток от деления r однозначно определяет номер страницы (процессора IOr), куда будет распределен кортеж. Процессоры верхнего уровня обработки запроса называются процессорами JOIN.

Есть еще три процессора. Процессор УПР реализует функции мониторинга и управления остальными процессорами системы. Процессор ПТР предназначен для претрансляции исходного запроса пользователя к виду регулярного дерева обработки запроса [7]. Для реализации функций объединения результатов обработки с уровня JOIN введен процессор SORT. Кроме объединения частных результатов на множестве узлов, этот процессор выполняет операции агрегации (реализации функций SUM(), AVG(), MAX(), MIN() и др.) и сортировки над результатом предшествующей операции соединения. Все три процессора функционируют на Host ЭВМ. В качестве языка запросов пользователя используется SQL. Обработка информации БД на низком уровне (работа на уровне файловой системы, системных буферов, алгоритмов доступа к данным, работы с индексами и т.п.) реализуется с помощью инструментальной СУБД MySQL. Доступ к функциям СУБД осуществ-

ляется через клиентское API (Application Program Interface) и системные библиотеки сервера MySQL. Взаимодействие через API подразумевает передачу SQL-запросов. Для обеспечения устойчивого функционирования программной системы использована барьерная синхронизация [12].

По условию пары IOr – JOINr функционируют на одном узле (конфигурация «линейка») SAN-кластера с параметрами: 22 узла 2 Quad-core Intel Xeon E5450 CPU/1,87GHz/32GB. Интерконнект между узлами – GigabitEthernet/Infiniband 4X (20Gbps DDR) с 24-портовыми коммутаторами Cisco. Дисковая подсистема узла – SAS диски XRB-SS2CD-146G10KZ с пропускной способностью 300 MB/s. В качестве представительского теста (ПТ) используется ограниченный тест TPC-H [13] из 14 запросов, не содержащих операций записи. Объем базы данных VБД = 5,4GB. Эффективность оценивается значениями двух параметров: 1/Tmin – обратная величина минимального времени обработки ПТ и hG – число узлов на грани масштабируемости, когда T = Tmin. Для снижения трудоемкости обработки запросов выполняется хеширование r-результатов обоих уровней по-отдельности между всеми IOr и между всеми Joinr по ключу очередного соединения. Итоговый результат от каждого IOr передается только «своему» Joinr, выполняющему операцию соединения со «своими» фрагментами хешированных данных.

Наличие двух процессоров в узле позволяет существенно улучшить масштабируемость «линейки» и снизить значение Tmin. Это достигается установкой на каждый SMP-узел вычислительного кластера двух серверов MySQL. Один из них связан с процессами IO, другой – с процессами Join. В итоге появляется еще одна конфигурация – «совмещенная симметрия». Для обеих конфигура ций проведена обработка последовательности запросов как конкатенации трех перестановок ПТ. Полученные результаты представлены на рис.1. (T – в сек.). Переход к конфигурация «совмещенная симметрия» показывает увеличение hG с 5 до 7 при снижении Tmin на 25% по сравнению с «линейкой».

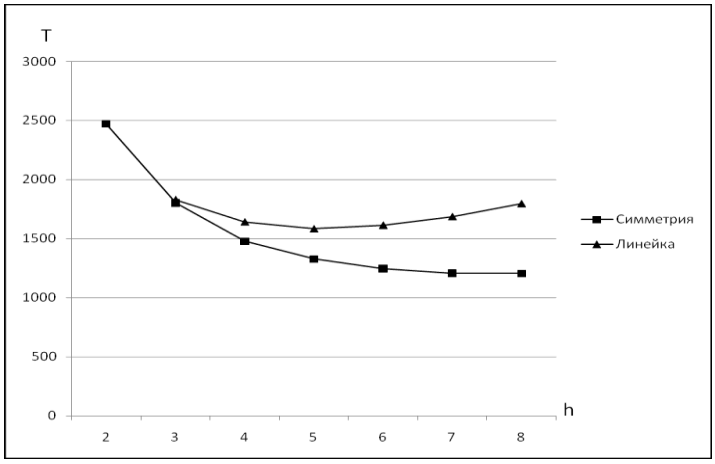

Рис. 1. Улучшение характеристик для симметрии (по оси абсцисс h – количество узлов, по оси ординат T – время выполнения пакета запросов в сек.)

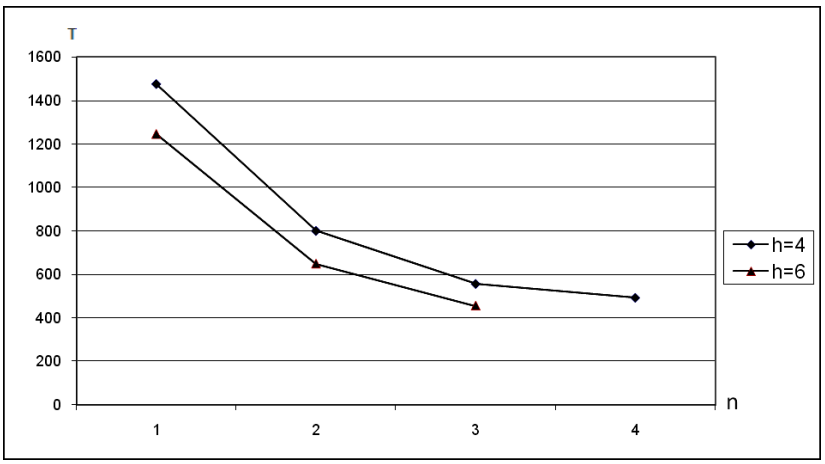

Рис. 2. Тестирование мультикластера (по оси абсцисс n – количество монокластеров, по оси ординат T – время выполнения пакета запросов в сек.)

Дальнейшее улучшение характеристик системы дает переход к мультикластеризации. При этом на Host дополнительно устанавливается программный модуль ROUTER, который распределяет поток запросов между n монокластерами-компонентами. Результаты проведенного тестирования при конфигурировании монокластеров-компонент как «совмещенная симметрия» и прежних условиях эксперимента показаны на рис.2. Варьировалось число узлов монокластера h. При h = 4 наблюдается рост масштабируемости мультикластера по сравнению с однокластерной системой в 2,3 раза; при h = 6 ≈ hG = 7 – более чем в 2,6 раз. Рост производительности на пороге масштабируемости – в 2,4 и не менее чем в 3 раза соответственно. Это подтверждает целесообразность выбора h = hG.

Установленное в [8] временное доминирование задержек барьерной синхронизации вблизи границы масштабируемости говорит о целесообразности развития асинхронного подхода к организации обработки запросов. При таком подходе простои в работе IOr и JOINr практически исключаются. Это должно существенно повысить эффективность мультикластера. В настоящее время проводятся исследования по организации асинхронного взаимодействия в алгоритмах обработки данных [14]. Частично асинхронная модель взаимодействия реализована в СУБД Greenplum [15]. Но особенности поведения асинхронных СУБД все еще мало изучены.

Разрабатываемый прототип асинхронной СУБД A’Clasterix использует наработки по СУБД Clasterix, но с существенными изменениями. Суть этих изменений состоит в следующем. В динамике работы системы для JOINr в общей памяти r-узла создаются буферные области для хранения фрагментов данных, получаемых от своего IOr. Сообщение, которое включает эти данные, дважды индексируется: номер запроса, номер отношения. Как только IOr закончит обработку r-фрагмента i-отношения j-запроса ([i,j]-отношения), он передает индексированное сообщение с полученными данными своему JOINr. Затем приступает к обработке r-фрагмента [(i+1), j)-отношения (возможно, 1-отношения [j+1]-запроса). Хеширование r-фрагмента промежуточного отношения теперь целесообразно выполнять на JOINr. О завершении этой работы он уведомляет Host. После чего читает очередное сообщение от IOr (если оно поступило) и приступает к новому хешированию. Host прерывает этот процесс по поступлении аналогичных уведомлений от всех других JOINr, приказывая всем JOINr начать операцию соединения с хешированием по-

лучаемых на них фрагментов временных отношений. По ее выполнении JOINr продолжает прерванное хеширование данных от своего IOr.

2. Один узел кластера на множество запросов

Использование многоядерных узлов позволяет по иному подойти к решению проблемы создания мощного узла на основе технологии MySQL. До сих пор самостоятельное применение СУБД MySQL не позволяло полностью использовать ресурсы многоядерных узлов кластера. Это являлось основной причиной сравнительно малой производительности и масштабируемости. В работе [9] удалось обеспечить 100% загрузку всех ядер при специальной настройке MySQL 5.6. Полученный при новых условиях эксперимента (ПТ – случайная последовательность из 800 запросов на множестве запросов ограниченного теста TPC-H) и определенной роутеризации в узле график зависимости времени выполнения ПТ в минутах от количества узлов в системе показан на рис.3. Наблюдается снижение Tmin на 25% в сравнении с конфигурацией «совмещенная симметрия» (рис.1) при неизменном hG.=7 и более чем на порядок большем числе обрабатываемых запросов. Дальнейшая мультикластеризация, аналогичная рассмотренной ранее, должна дать прежний дополнительный эффект.

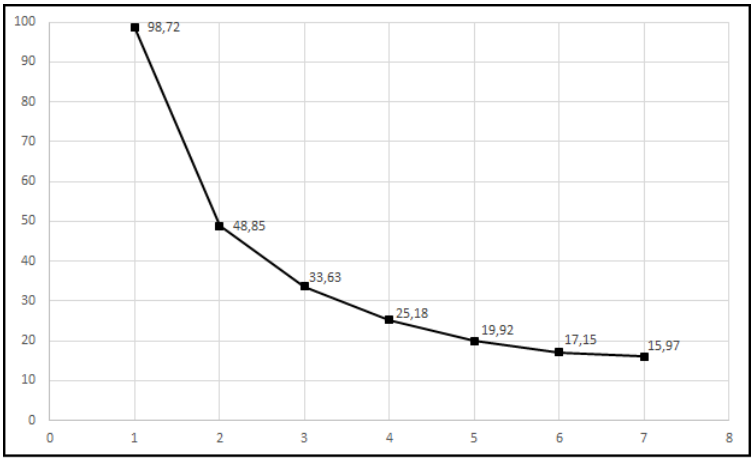

Рис. 3. График среднего времени выполнения ПТ (по оси абсцисс – количество узлов, по оси ординат – время выполнения пакета запросов в мин.)

Заключение

Обе рассмотренные стратегии используют инструментальное средство MySQL с исследовательскими версиями дополнительной программной надстройки. Но во втором случае эта надстройка намного проще, и довести ее до конечного программного продукта не составит затруднений. Стратегия «множество запросов на один узел» предпочтительна для практики и потому, что обеспечивает существенно более высокую эффективность. Дальнейшее развитие исследований по первому сценарию связывается с переходом на асинхронность. Не исключено, что благодаря исключению барьерной синхронизации стратегия «один запрос на множество узлов» приблизится по эффективности ко второму варианту. Каковы будут сравнительные результаты, покажет будущее. Первый вариант представляет особый интерес тем, что горизонтальное деление базы данных позволяет работать с БД бóльших объемов.

Литература

1. Xu, Y. Handling data skew in parallel joins in shared-nothing systems /Y. Xu, P. Kostamaa, X. Zhou, L. Chen //ACM SIGMOD international Conference on Management 9of Data Canada, 2008, proceedings. ACM, 2008. – P.1043-1052.
2. Лепихов, А.В. Параллельная обработка запросов в СУБД для кластерных вычислительных систем /А.В. Лепихов //Отчет в рамках гранта МК-3535.2009.9.
3. Shiers, J. The Worldwide LHC Computing Grid (worldwide LCG) /J. Shiers //Computer Physics Communications. – 2007. – Vol. 177, No. 1-2. – P.219–223.
4. Szalay, A.S. The Sloan Digital Sky Survey and beyond /A.S. Szalay//SIGMOD Record. – 2008. – Vol. 37, No. 2. – P.61–66.
5. Абрамов, Е.В. Параллельная СУБД Clusterix. Разработка прототипа и его натурное исследование /Е.В. Абрамов //Вестник КГТУ им. А.Н. Туполева. – 2006. – №2. – С.50-55.
6. Райхлин, В.А. Информационные кластеры как диссипативные системы /В.А. Райхлин, Д.О. Шагеев //Нелинейный мир. – 2009. – Т.7, №5.– С.323-334.
7. Raikhlin, V.A. Simulation of Distributed Database Machines /V.A. Raikhlin //Programming and Computer Software. – 1996. – Vol. 22, Issue 2. – P.68-74.
8. Райхлин, В.А. Мультикластеризация распределенных СУБД консервативного типа /В.А. Райхлин, Р.Ш. Минязев //Нелинейный мир. – 2011. – №8. – С.473-481.
9. Классен, Р.К. Повышение эффективности параллельной СУБД консервативного типа на кластерной платформе с многоядерными узлами /Р.К. Классен //Вестник КГТУ им. А.Н. Туполева. – 2015. – №1. – С.112-118.
10. Отчет о НИР. Развитие принципов построения процессоров массивов с широкими функциональными возможностями как внешних устройств ЭВМ ЕС. Адаптация процессора массивов к макроконвейерной системе /Научн. рук. В.А. Райхлин //Гос.рег. № 01850074085. Инв. № 02860096253. – Казань, 1986.
11. Мартин, Дж. Организация баз данных в вычислительных системах. Издание второе, дополненное. Перевод с английского /Дж. Мартин – М.: Мир, 1980. – 662 с.
12. Воеводин, В.В. Параллельные вычисления /В.В. Воеводин, Вл.В. Воеводин – СПб.: БХВ-Петербург, 2004. – 608 с.
13. TPC Benchmark™ H Standard Specification Revision 2.17.1, URL:
14. http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf
15. Zhu, YingHui. Study on Parallel Clustering Based on Asynchronous Communication /YingHui Zhu, YuZhen Jiang, Bo Liu //International Conference on Networking and Digital Society. – 2009. – Vol. 2. – P.24-27.
16. Cohen, Jeffrey. MAD Skills: New Analysis Practices for Big Data /Jeffrey Cohen, Brian Dolan, Mark Dunlap, Joseph M. Hellerstein, Caleb Welton. //Proceedings of theVLDB'09 Conference. – 2009. – P.1481-1492.