УДК 004.031.43

УВЕЛИЧЕНИЕ ПРОПУСКНОЙ СПОСОБНОСТИ БАЗ ДАННЫХ В ОБЛАСТИ СОТОВОЙ СВЯЗИ

Божко К.А., аспирант ФГУП «Научно-исследовательский центр информатики при Министерстве иностранных дел Российской Федерации»

В статье представлены результаты экспериментального моделирования, проведенного с целью сравнительной оценки преимуществ использования резидентных в оперативной памяти баз данных перед стандартной СУБД для увеличения пропускной способности при их применении в области сотовой связи.

Ключевые слова: информационная система, абонент, сотовая связь, система управления базами данных, модель.

UDC 004.031.43

INCREASING THE CAPACITY OF CELLULAR NETWORK DATABASES

Bozhko K.A., postgraduate student at FSUE «Research and Development Centre on Informatics by the Ministry of Foreign Affairs of the Russian Federation»

The article provides the result of experimental modeling for comparative assessment of the advantages of usage of RAM resident databases versus standard DBMS in order to increase their capacity when he used in cellular networks.

Keywords: information system, subscriber, sell the network, database management system, model.

Для удовлетворения быстро растущих требований к информационным технологиям от бизнеса необходимо рассматривать альтернативные методы построения элементов информационных систем. В последние годы приобрёл популярность новый класс баз данных – резидентные в оперативной памяти базы данных (in-memory databases). Основная особенность архитектуры таких систем управления базами данных (СУБД) состоит в том, что база данных целиком размещается в оперативной памяти сервера, что позволяет отказаться от достаточно сложных алгоритмов обслуживания кэш-буферов традиционных

СУБД, а это, в свою очередь, даёт существенный выигрыш в производительности приложений баз данных.

Разработчики этих СУБД позиционируют их как СУБД реального времени, поэтому резидентные в оперативной памяти базы данных используются только в специальных областях применения, требующих чрезвычайно малого времени отклика на запросы к базе данных, таких как обслуживание оборудования для мобильной связи, биржевых торгов и в некоторых других областях применения. Заявляется, что производительность резидентных в оперативной памяти баз данных в десятки раз превосходит производительность традиционных реляционных баз данных.

В данной статье представлены опытные показатели сравнения производительности традиционной реляционной СУБД – сервера Oracle и СУБД Oracle TimesTen, работающей с резидентными в оперативной памяти базами данных.

Для современной мобильной связи актуально использование высокопроизводительных систем регистрации длительности звонков, а также использования абонентами дополнительных услуг связи, например, роуминг, GPRS (General Packet Radio Service – пакетная радиосвязь общего пользования), переадресация вызова.

Стандартная система такова: вся информация сохраняется напрямую в одной таблице в центральной, «дисковой» реляционной базе данных, где потом обрабатывается и используется приложениями. Однако с ростом количества абонентов увеличивается число обращений к таблице. Как следствие, растёт конкуренция между обращениями. При достижении максимальной производительности увеличивается время ожидания в очереди за доступом к таблице.

Использование резидентных в памяти баз данных предлагает альтернативное решение для описанной системы – задействовать набор таких баз в качестве посредников между приложением, обслуживающим звонки, и цен-

тральной «дисковой» СУБД. Иначе говоря, предлагается децентрализовать процесс сбора информации по звонкам, распределив несколько экземпляров регистрирующего приложения и резидентных оперативной в памяти баз данных на отдельных узлах по всей зоне обслуживания абонентов [1].

На каждом отдельном узле приложение соединяется локально с резидентной в оперативной памяти базой для регистрации начала и/или окончания вызова в своей географической зоне. Причём принципиальным моментов в этой схеме является нахождение базы и приложения в пределах одного физического сервера. В противном случае значительную лепту в ухудшение производительности вносят накладные расходы, связанные с передачей информации по сети. Заявляется, что такая схема позволит увеличить пропускную способность, а время отклика уменьшить примерно в 20 раз. Другими словами, система сможет обслужить за единицу времени в 20 раз больше абонентов.

Для каждого вызова, приложение регистрирует отдельно и время начала, и время окончания звонка. Это необходимо для тех случаев, когда начало вызова регистрируется одним узлом, а окончание другим.

Для передачи данных в центральную СУБД на каждом узле должен быть поднят фоновый процесс, который соединяется локально с экземпляром базы данных «в памяти» и удалённо с центральной.

Целью проведённых экспериментов была количественная оценка преимуществ использования резидентных в оперативной памяти баз данных перед стандартной СУБД. Основным показателем являлось максимальная производительность приложения – то есть такое максимальное количество сгенерированных вызовов, которое приложение способно обработать без потерь на данном сервере. Для этого сначала генерировалось небольшое количество вызовов в секунду, после чего эта величина увеличивалась до тех пор, пока будет достигнута верхняя граница производительности приложения.

Также обе схемы (с использованием стандартной СУБД и резидентной в оперативной памяти базой данных) тестировались в двух режимах записи

журнальной информации на диск – синхронном и асинхронном. Это делалось потому, что использование резидентных в оперативной памяти баз данных в своей основе предполагает асинхронную, фоновую запись на диск журнальной информации, в то время как такое использование нехарактерно для стандартной СУБД.

В качестве альтернативы двум указанным режимам проводился опыт с использованием репликации встроенной в оперативную память базы данных на аналогичный сервер. В такой схеме обеспечивалась согласованность данных в случае остановки основного экземпляра встроенной в оперативную память базы данных. Такую схему можно сравнивать с синхронным режимом записи журнальной информации на диск с использованием стандартной СУБД.

После получения показателей было необходимо провести их сравнение для трёх различных случаев, так как сами по себе они характеризуют не столько систему, сколько сервер, на котором проводился эксперимент. А отношения этих показателей уже дают количественные характеристики, позволяющие оценить выигрыш в производительности при использовании баз данных «в памяти» не зависимо от сервера.

Хотелось максимально приблизить эксперимент к процессам, происходящим в реальной жизни. С этой целью для реализации вызовов абонентов был создан виртуальный генератор. Эксперимент представляет собою макет, генерирующий звонок абонента по мобильной связи и обрабатывающий этот вызов. Схема эксперимента состояла из трёх частей: генератор, приложение и база данных.

Генератор в данной схеме заменял датчики реального оборудования. Он с заданным математическим ожиданием количества звонков в секунду создаёт вызовы. Для связи с реальной жизнью звонки создавались, основываясь на генераторе случайных чисел, выполненном с использованием функции rand ().

Если задать сгенерированное в секунду количество случайных чисел как n, а вероятность того, что отдельное число будет соответствовать возникновению события, принять за p, то математическое ожидание количества звонков в секунду равно $M = p \cdot n$; а отклонение от него описывается распределением Пуассона.

Генератор и приложение взаимодействовали посредством сегментов разделяемой памяти. Такой способ называется механизм взаимодействия IPC. Он обеспечивает наиболее быстрый обмен данными между различными процессами в системе. При возникновении события генератор делает запись в ячейке разделяемого сегмента памяти и переводит курсор на следующую ячейку. Таким образом, в разделяемой памяти создаётся очередь вызовов, которая потом обрабатывается приложением.

Приложение подключается к базе данных и начинает читать очередь вызовов в разделяемом сегменте памяти. Если в системе генерируется звонок, приложение вставляет запись в таблицу в базе данных, после чего переходит к следующей ячейке разделяемого сегмента. Принципиальным моментом было проверить, что без вставки в таблицу количество обработанных звонков значительно выше, чтобы показатели производительности характеризовали именно базу данных, а не определённый сервер или операционную систему.

Для случая с использованием стандартной СУБД (СУБД Oracle11g) приложение обращалось к ней с использование интерфейса OCI (Oracle Call Interface – специализированный интерфейс доступа к базам данных Oracle), а в случае с резидентной в оперативной памяти базы даны (TimesTen 11.2.2) – интерфейса ODBC (Open Database Connectivity - программный интерфейс доступа к базам данных). Это делалось для того, чтобы сравнивать указанные базы в равных условиях – использованные интерфейсы являются для них наиболее производительными соответственно.

Ниже приведены графики сравнения производительности для трёх схем (рис. 1 и рис. 2).

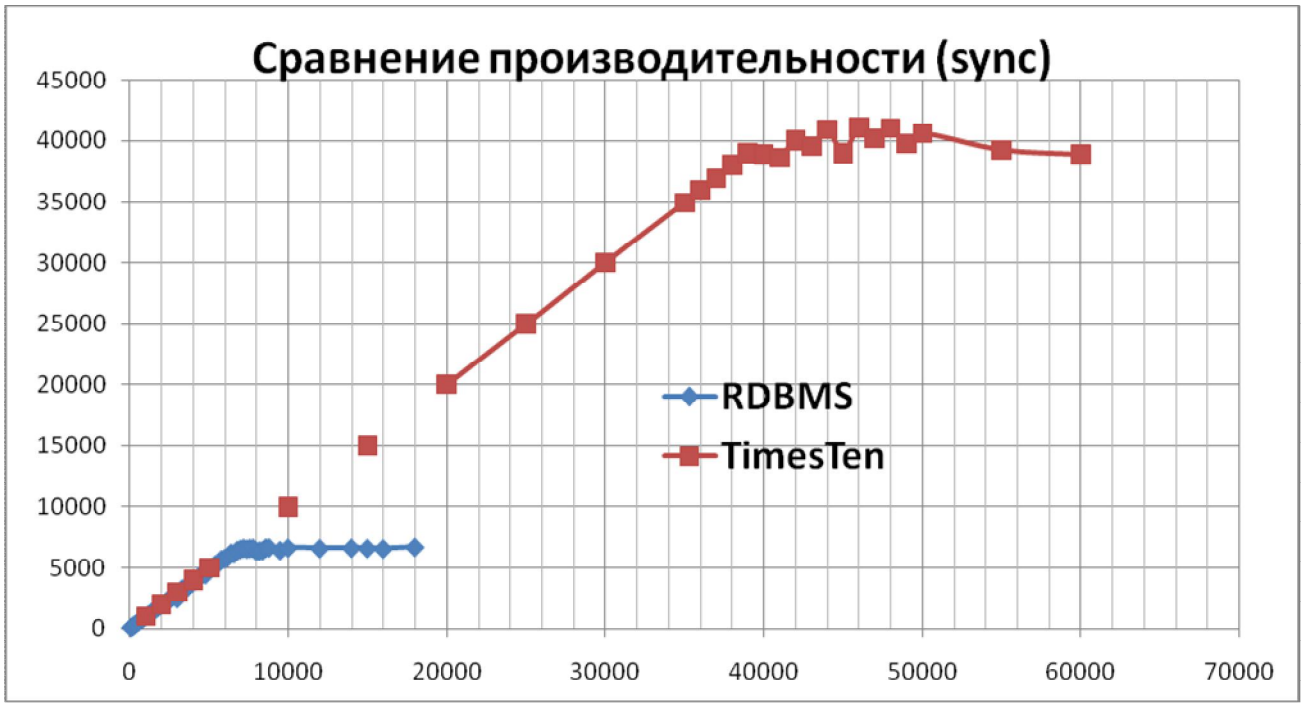

Рис. 1. Сравнение производительности в синхронном случае

Из графика на рис. 1 видно, что TimesTen увеличивает производительность системы почти в 6 раза в случае согласованной записи журнальной информации на диск.

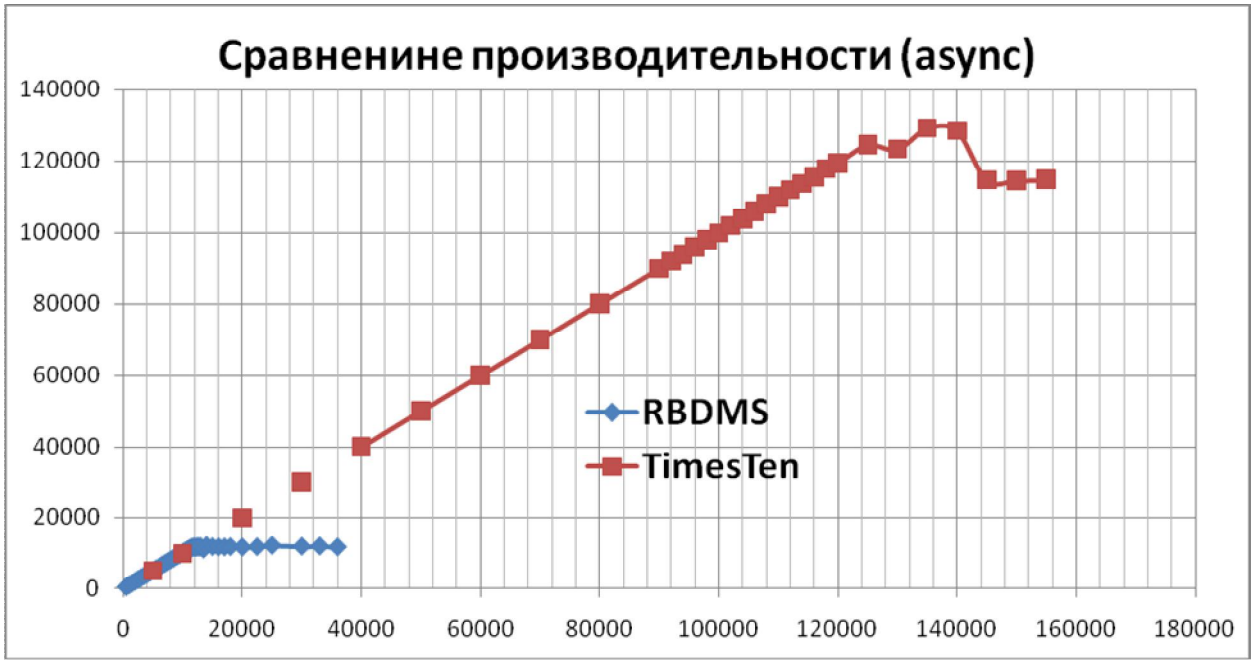

Рис. 2. Сравнение производительности в асинхронном случае

Синхронный случай в данном эксперименте означает, что приложение не может продолжать работу, пока не завершится запись информации в журнальные файлы, асинхронный – наоборот.

В асинхронном случае (рис. 2) использование TimesTen увеличивает производительность до 10 раз. Но в этом случае при аварийной остановке базы данных (обоих классов) существует вероятность потери части информации, которая не записана на диск. В случае репликации такая вероятность отсутствует.

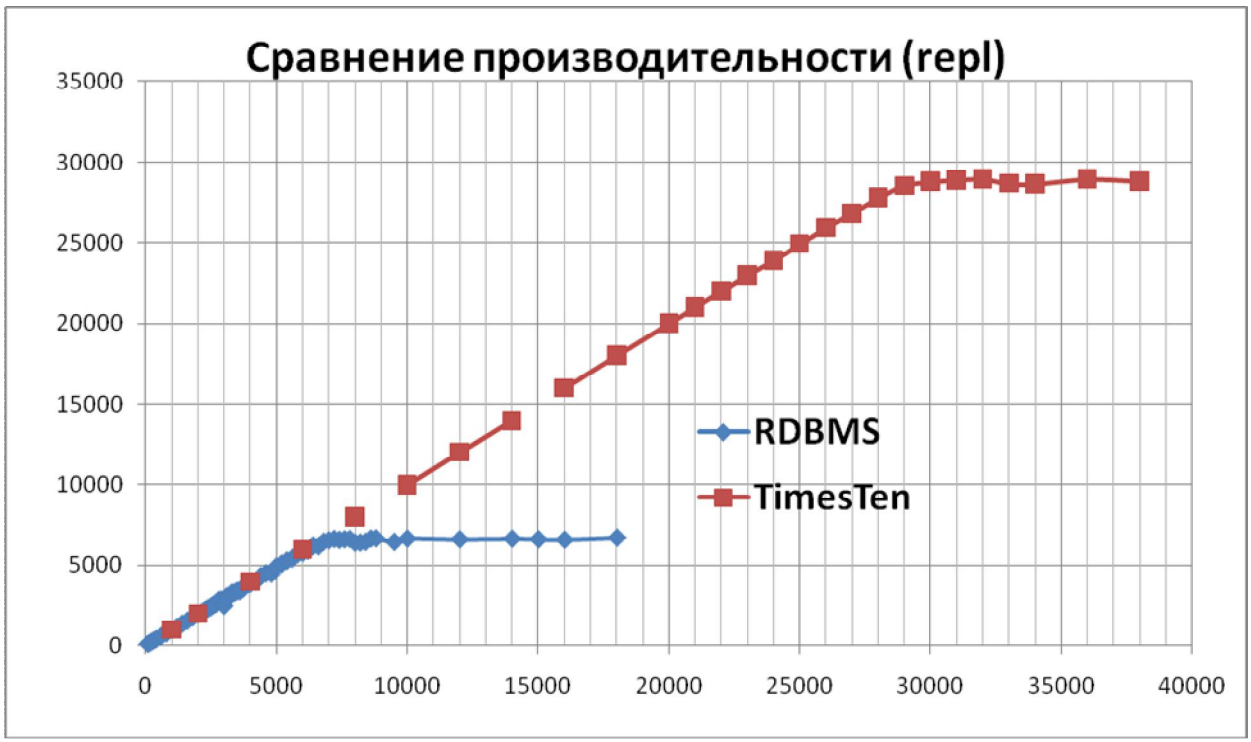

Рис. 3. Сравнение производительности для случая репликации

В случае репликации (рис. 3) использование TimesTen увеличивает производительность до 5 раз.

Из проведённых экспериментов можно сделать следующие выводы: во-первых, использование резидентных в оперативной памяти в памяти баз данных в системах регистрации звонков мобильной связи более эффективно по сравнению со стандартными, «дисковыми» СУБД – они позволяют значительно, до двадцати раз, увеличить производительность системы. Это означает, что за единицу времени большее количество абонентов смогут воспользовать-

ся услугами оператора сотовой сети, во-вторых, оно даёт значительное уменьшение времени отклика системы. Хотя на данный момент большая часть времени соединения тратится на работу оборудования, обеспечивающего сервис мобильной связи, а не регистрацию звонка, этот показатель не будет являться узким местом при соединении абонента ещё долгое время.

Однако нужно отметить, что резидентные в оперативной памяти базы данных имеют и обратную сторону медали. Во-первых, на их производительность значительно влияет обращение к диску – характерно для синхронного случая. В проведённом эксперименте завершение транзакции (commit) выполнялся после 100 вставок в таблицу. Если же выполнять завершение транзакции после каждой вставки, то показатели резидентной в памяти базы оказываются значительно ниже, чем при использовании стандартной СУБД. То есть в таком случае преимущество проявляется только в асинхронном эксперименте.

Во-вторых, производители резидентных в оперативной памяти баз данных рекомендуют использовать для обеспечения надёжности технологию репликации – повторения всей или части информации на другом сервере. Однако сразу настораживает тот факт, что накладные расходы сети серьёзно ухудшают показатели производительности.

С экономической точки зрения использование этого метода является крайне эффективным, так как позволяет увеличить производительность системы до 10 раз, сократить время доступа к информации и обеспечить необходимую надёжность за счёт дублирования информации.

Литература

1) Документация Oracle. «Oracle TimesTen In-Memory Database Introduction Release 7.0». http://download.oracle.com/otn_hosted_doc/timesten/701/TimesTen-Documentation/intro.pdf

2) Alex Otwagin. «Введение в ОС Linux. Разделяемые сегменты памяти». http://skif.bas-net.by/bsuir/base/node233.html.

3) Документация Oracle. «Oracle® Call Interface Programmer's Guide». http://docs.oracle.com/cd/E11882_01/appdev.112/e10646.pdf.

4) Документация Oracle. «Oracle TimesTen In-Memory Database C Developer’s Guide 11g Release 2 (11.2.2)». http://docs.oracle.com/cd/E21901_01/doc/timesten.1122/e21637.pdf.