SYBASE®
An SAP® Company

ТЕХНИЧЕСКИЙ ДОКУМЕНТ

Replication Server, выпуск 15.5

Рекомендации по настройке и оптимизации быстродействия

СОДЕРЖАНИЕ

Резюме

Выпуск 15.5 программы Replication Server (RS) обеспечивает повышенное быстродействие и масштабируемость для систем репликации, используемых при решении ответственных задач. В настоящем документе рассмотрены основные улучшения программы в части быстродействия, а также измененный порядок ее настройки на оптимальную производительность.

Основное нововведение в RS 15.5 — механизм высокообъемной адаптивной репликации (High Volume Adaptive Replication — HVAR), применяемый для репликации больших объемов данных как в ASE, так и в IQ. HVAR существенно оптимизирует процесс репликации реального времени в IQ (данная функция доступна в редакции RS 15.5 RTL).

Испытания, проведенные в нашей лаборатории, показали, что RS 15.5 превосходит предыдущие выпуски программы в быстродействии как при использовании в качестве приемников ASE, так и IQ.

- В системах онлайновой обработки транзакций (OLTP) при репликации из ASE в ASE достигнут более чем 200-процентный прирост производительности.
- Достигнуто более чем 600-процентное повышение быстродействия в реальных приложениях, в транзакционном профиле которых доминируют команды вставки как таковые.
- Реализована загрузка в реальном времени в приемники IQ из источников OLTP с помощью Sybase Replication.
- Реализована поддержка большого числа одновременно работающих пользователей на многопроцессорных системам симметричной архитектуры с полной их загрузкой.
- Достигнуто значительное сокращение объема операций по созданию устройств и настройке репликации.

В RS 15.5 (редакция RTL) впервые реализована возможность непосредственной, то есть без промежуточных ступеней, репликации OLTP-транзакций из ASE в IQ в реальном времени. Благодаря этому структура систем репликации с использованием IQ значительно упрощается.

1. Введение

Sybase Replication Server — это программное обеспечение репликации транзакций в реальном времени и средство распределения данных, ставшее отраслевым стандартом. Программа обеспечивает пересылку и синхронизацию данных в масштабе территориально-распределенных организаций. Replication Server позволяет решать проблемы управления корпоративной информацией, такие как репликация и синхронизация гетерогенных данных, отчетность в реальном времени, обеспечение высокой готовности и аварийного восстановления. Это облегчает организациям задачи соблюдения постоянно возрастающих отраслевых требований, снижения риска функционирования территориально-распределенных информационных систем, а также эффективного использования ИТ-активов, достающихся при слияниях и поглощениях.

Назначение настоящего документа — отразить различные сведения, касающиеся быстродействия и дать читателю понимание разных методов оптимизации работы RS 15.5. Здесь будут приведены результаты различных тестов программы на быстродействие. Они должны послужить администраторам RS в качестве указаний. Некоторые из этих указаний применимы не только к выпуску 15.5, но и к более ранним версиям Replication Server. Цель нашей работы — дать пользователям RS действенное средство быстрого разрешения проблем недостаточного быстродействия, которое помогло бы им сберечь время и силы.

Документ адресован следующим кругам читателей:

- пользователям, желающим узнать о введенных в RS 15.5 возможностях повышения быстродействия и их применимости к конкретным задачам. Следует подчеркнуть, что авторы не ставили целью дать здесь исчерпывающее описание каждой новой возможности выпуска 15.5 — за этой информацией следует обратиться к документу «What's New In Replication Server Version 15.5»;
- администраторам БД, желающим настроить свои системы RS System на максимальное быстродействие;
- в качестве вспомогательного руководства — администраторам БД, которые столкнулись с проблемой низкой скорости репликации и желают найти и устранить ее причину.

Предполагается, что читатель уже знаком с устройством и работой Replication Server, а также владеет основными терминами (вся необходимая информация приведена в руководствах по работе с программой). В основном будет рассмотрено применение Replication Server в среде Adaptive Server Enterprise, хотя в обзор включен и раздел по IQ в роли базы-реплики.

2. Спецификация системы

Описанные испытания проводились на системе следующей конфигурации:

Сервер	HP Proliant DL580
ЦП	16 ядер — Intel® Xeon® CPU X7350 @ 2,93 ГГц
ОЗУ	32 Гбайт
ОС	Red Hat Enterprise Linux Server версии 5.2 (Tikanga) Linux 2.6.18-92.el5 #1 SMP x86_64 x86_64 x86_64 GNU/Linux
Диск	EMC Symmetric Disk Array Rev: 5874 Тип: прямой доступ
ПО	Replication Server 15.5 (64-разрядный), Adaptive Server Enterprise 15.0.3 (64-разрядный)

3. Проведенные испытания

Испытания, результаты которых приведены в настоящем документе, проводились с OLTP-приложением, работающим с серверами ASE 15.0.3 и RS 15.5 на базе операционной системы Linux. Выбор OLTP-приложения объясняется тем, что это наиболее распространенный в практике тип бизнес-приложений, для поддержки которых используется Replication Server. Кроме того, в некоторых случаях проводился тест, состоящий исключительно из операций вставки больших и малых порций данных вразбивку. ASE-источник был установлен на одной машине, а ASE-приемник и сам Replication Server — на другой (в таблице 1 приведена спецификация машины для Replication Server). Репликация данных выполнялась в режиме MSA путем создания определений репликации и подписок для всех таблиц базы-источника. В ASE-источнике в целях считывания и пересылки журнала был установлен RepAgent/ASE (Replication Agent Thread, или RAT).

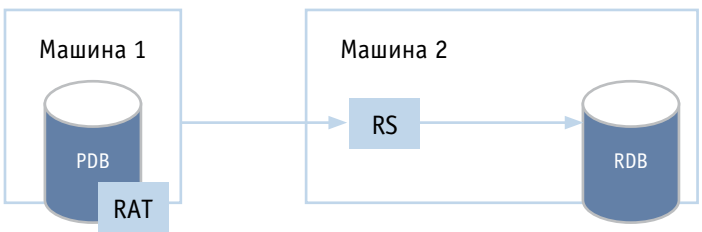

Рис. 1

4. Нововведения в RS 15.5

Выпуск RS 15.5 рассчитан на обеспечение высокого быстродействия. Высокообъемная адаптивная репликация (High Volume Adaptive Replication, HVAR) — это патентуемая технология, направленная на существенное ускорение репликации в OLTP-системах с большим объемом данных. В предыдущих версиях Replication Server каждая операция (вставка, удаление или изменение) непосредственно транслировалась в базу-приемник, запись за записью и в порядке, соответствующем журналу — в режиме непрерывной репликации (за исключением репликации хранимых процедур и команд SQL). Реализованная в RS 15.5 технология HVAR основана на компиляции и массовом применении операций — в результате объем передаваемых данных уменьшается, а скорость, по сравнению с режимом непрерывной репликации, увеличивается.

- В процессе компиляции выполняется группировка операций по таблицам, а также по командам вставки, изменения и удаления таким образом, чтобы из нескольких команд, изменяющих одну запись, осталась только одна, формирующая конечный результат.
- Итоговый результат компиляции помещается в приемник в режиме массовой загрузки — это самый быстрый способ загрузки (задача облегчается благодаря применению уже упомянутого механизма группировки на этапе компиляции). Он намного более эффективен, чем пересылка изменений каждой записи отдельно.

Итоговые изменения записей, применяемых к базе-приемнику, Replication Server хранит в специальной БД в оперативной памяти. Компиляция исключает необходимость пересылки каждой зафиксированной в журнале операции — как уже было отмечено, все промежуточные операции исключаются и в приемник передается только окончательный результат реплицируемой транзакции. HVAR особенно эффективна в системах OLTP-архивирования и отчетности, где базы приемники имеют схемы, аналогичные схемам баз-источников.

К числу других заметных усовершенствований RS 15.5 в части оптимизации относятся следующие (многие из них касаются исключительно внутреннего устройства продукта, «прозрачны» для пользователя и могут быть не документированы):

- Модуль Distributor теперь выполняет считывание непосредственно из кэша Stable Queue Transaction.
- Исключены большие накладные расходы по обработке цепей LRU для полностью кэшируемых таблиц RSSD.
- Уменьшено количества выделений и высвобождений блоков благодаря увеличенному размеру блока SQM.
- Параллелизованы процессы разбора и нормализации модуля EXEC путем разделения его на два потока.
- Исключены излишние операции планирования SMP.
- Уменьшена конкуренция за внутренние ресурсы, блокируемые потоками.
- Усовершенствован модуль DSI: исполнение сервером БД текущей команды перекрывается с подготовкой в Replication Server следующей команды, подлежащей передаче серверу БД.
- Исключены циклические блокировки путем реализации ассемблерных атомарных команд добавления/удаления для активно используемых переменных.

На рис. 2 приведены некоторые результаты тестов, проведенных в лаборатории.

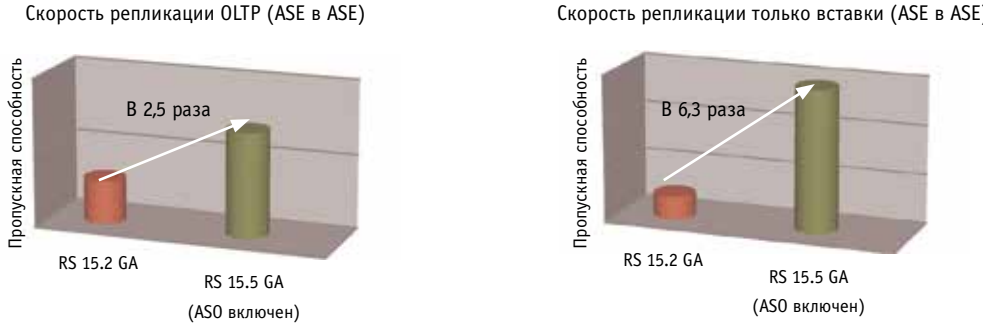

Рис. 2

Помимо перечисленных усовершенствований, в RS 15.5 были оптимизированы некоторые аспекты ввода-вывода, благодаря чему удалось значительно сократить время запуска постоянных устройств.

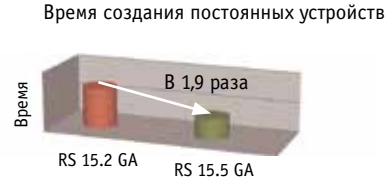

Рис. 3

В дальнейшем изложении приведены различные советы по анализу быстродействия для выявления узких мест при репликации как из ASE в ASE, так и из ASE в IQ. Предлагаются способы подстройки Replication Agent/ASE, Replication Server и приемников ASE или IQ в целях повышения быстродействия; также представлены новые средства достижения быстродействия в RS 15.5.

5. Сервер — источник данных

Когда речь идет о репликации, главным представляющим интерес компонентом сервера — источника данных является Replication Agent.

5.1. Анализ быстродействия Replication Agent

В ASE агент Replication Agent реализован как задача ASE. Этот поток просматривает журнал, в котором регистрируются пользовательские транзакции, преобразует его в формат LTL (Log Transfer Language) и пересылает на Replication Server. В последующих параграфах подробно описаны некоторые из распространенных проблем недостаточного быстродействия, могущие возникать в работе Replication Agent, а также приведены рекомендации по настройкам ASE и системы в целом, позволяющие указанные проблемы решать.

5.1.1. Коммуникация с Replication Server

Анализируя быстродействие Replication Agent, в первую очередь следует обратить внимание на то, сколько времени ему приходится ждать ответа от Replication Server. Эти данные можно найти в разделе «Replication Agent» отчета sysmon ASE, вкладка «I/O waits from RS».

Таблица 2

	per sec	per xact	count	% of total
	---	---	---	---
I/O Wait from RS				
Count	n/a	n/a	23142	n/a
Amount of Time (ms)	n/a	n/a	69723	n/a
Longest Wait (ms)	n/a	n/a	296	n/a
Average Wait (ms)	n/a	n/a	3.0	n/a

В проведенном тесте (см. табл. 2) Replication Agent потребовалось около 82 секунд на то, чтобы считать журнал, создать LTL и отправить его RS. Как видно из таблицы, на ожидание ответов Replication Server в совокупности ушло около 70 секунд.

Для сокращения времени ожидания агентом Replication Agent ввода-вывода следует рассмотреть три аспекта:

1) время обработки данных сервером Replication Server (модулем EXEC);

2) пропускная способность сети между Replication Agent и Replication Server;

3) планирование задач в ASE.

Способы повышения быстродействия модуля EXEC будут подробно рассмотрены в разделе 6.4. Планирование задач в ASE рассматривается в параграфе 5.1.3, а устранение задержек при передаче данных по сети описывается в параграфе 5.1.4. В данном разделе рассматриваются некоторые способы оптимизации взаимодействия Replication Agent и Replication Server.

Число обменов данными между Replication Agent и Replication Server в первую очередь определяется новым параметром настройки Replication Agentltl batch size. Этот параметр задает максимальный объем пакета данных LTL, который Replication Agent пересылает Replication Server. Значениеltl batch size может варьировать от 16 384 до 2 147 483 647 байт. Значение по умолчанию — 16 384. В версиях ASE младше 15.0.3 это значение всегда равнялось 16 384, изменять его было нельзя. Быстродействие Replication Agent можно улучшить, увеличив размер пакета LTL. В конце передачи каждого пакета Replication Agent проверяет корректность передачи, таким образом, увеличение размера пакета сокращает число проверок. В случае наличия ошибок это может увеличить объем работы, который Replication Agent придется проделывать повторно, однако если ошибок нет, то увеличение этого параметра может быть весьма полезным, в особенности если Replication Agent и Replication Server территориально разнесены или скорость передачи данных в сети мала. Диаграмма ниже иллюстрирует отмеченное в испытаниях уменьшение показателя «I/O waits from RS» по мере увеличения параметраltl batch size.

Рис. 4

5.1.2. Затраты времени на считывание журнала

Приведенная в таблице 3 часть раздела Replication Agent отчета sysmon содержит время, затраченное Replication Agent на считывание журнала.

Таблица 3

	per sec	per xact	count	% of total
	----------	----------	----------	----------
	Log Scan Summary			
Log Records Scanned	n/a	n/a	4244984	n/a
Log Records Processed	n/a	n/a	2693452	n/a
Number of Log Scans	n/a	n/a	1	n/a
Amount of Time for Log Scans (ms)	n/a	n/a	3523	n/a
Longest Time for Log Scan (ms)	n/a	n/a	3523	n/a
Average Time per Log Scan (ms)	n/a	n/a	3523.0	n/a

Если оказывается, что времени на считывание журнала уходит много и в отчете sysmon ASE для соответствующего устройства, где хранится журнал транзакций, зарегистрированы физические операции чтения, следует попытаться увеличить или настроить кэш журнала, как показано ниже.

sp_cacheconfig 'logcache', '<size>', logonly

go

sp_bindcache logcache, '<pdb>', syslogs

go

Replication Agent должен постоянно считывать страницы журнала. Текущая страница журнала транзакций записывается на диск при фиксации транзакции. Целью изменения размера кэша журнала транзакций является не исключение записи, а сокращение числа обращений Replication Agent к диску для чтения выгруженных из кэша страниц. Таким образом, уменьшается число операций считывания с диска, на которые уходит продолжительное время. При определении размера кэша для журнала транзакций следует начать с небольшого значения и увеличивать его до тех пор, пока не прекратится чтение с устройства, где хранится журнал транзакций основной базы данных.

Примечание: сервер может периодически испытывать значительные скачки нагрузки по пакетной обработке, и в этом случае может потребоваться кэширование значительно большего объема журнала транзакций (иногда может даже не хватать памяти на кэширование всех страниц журнала). Кэш, настроенный из расчета на такую нагрузку, может использоваться не полностью.

5.1.3. Планирование задач ASE

Отмечено, что когда ASE нагружает ЦП более чем на 90%, быстродействие Replication Agent существенно снижается. Одна из причин этого связана с использованием Replication Agent для связи с Replication Server режима отложенного ввода-вывода CT-lib. Заметим, что после выдачи каждой команды ввода-вывода CT-lib задача Replication Agent в ASE должна возвратить результат. Завершение операции ввода-вывода проверяется с помощью вызова ct_poll() из контекста планировщика. Чем сильнее механизм ASE загружает ЦП, тем больше задержка перед тем, как планировщик ASE получит возможность запросить о завершении операции ввода-вывода CT-lib.

Начиная с версии ASE 15.0.3 (а также в выпуске 12.5.4 ESD#8) добавлен новый параметр настройки Replication Agent bind to engine, позволяющий привязать Replication Agent к определенному экземпляру механизма ASE. Он позволяет, если есть такая возможность, зарезервировать один из экземпляров механизма исключительно для Replication Agent, а пользовательские приложения привязать к остальным экземплярам. Поскольку экземпляр механизма свободен от выполнения других задач ASE, планировщик может запрашивать о завершении операции ввода-вывода CT-lib значительно чаще.

Параметр bind to engine полезен также, когда в одном сервере ASE запущено несколько потоков Replication Agent, и администраторы БД должны вручную заниматься балансировкой нагрузки, распределяя эти агенты по разным экземплярам механизма.

Другая проблема, связанная с планированием задач, может возникнуть тогда, когда Replication Agent является единственной задачей, запущенной на ASE. Переслав данные серверу Replication Server, задача Replication Agent переходит в режим ожидания ответа. Планировщик ASE, обнаружив отсутствие подлежащих счету задач, может, после оговоренного числа проверок их наличия (это число задается конфигурационным параметром ASE Runnable Process Search Count), высвободить процессор. Если ответ на запрос Replication Agent поступит в этот период, агент рискует его не получить до тех пор, пока механизмы ASE не начнут свою работу снова. При этом быстродействие Replication Agent может снизиться. Один из способов решить проблему — попытаться подобрать подходящее значение параметра Runnable Process Search Count.

5.1.4. Настройка сети обмена данными между Replication Agent и Replication Server

Если установлено большое значениеltl batch size (как правило, больше 64K), Replication Agent будет пересылать Replication Server большие пакеты данных. Однако пересылаемые данные буферизуются на уровне TCP в машине-источнике и занимаемую ими память нельзя освобождать до тех пор, пока от машины-приемника не получено подтверждение приема. Если размер окна (буфера), используемый на уровне TCP, мал (значение по умолчанию в Linux — 16K), может случиться, что когда Replication Agent попытается переслать очередной TCP-пакет, буфер будет полон. В этом случае отправка пакета будет задержана до тех пор, пока компонент уровня TCP не получит подтверждения доставки ранее отправленных TCP-пакетов и не высвободит память. Задержка в отправке TCP-пакетов повлечет задержку в получении подтверждения, а также ответа от Replication Agent в конце передачи пакета данных. В случае если скорость сетевого обмена между Replication Agent и Replication Server слишком мала (10 Мбит/с или меньше, например, если Replication Agent и Replication Agent территориально разнесены), быстродействие может существенно снизиться. Время задержки при отправке командных пакетов можно увидеть с помощью raw-счетчика Replication Agent ra_sum_io_send_wait.

Таблица 4

field_name	group_name	field_id	value
----------	----------	----------	----------
ra_io_send	repagent_4	7	17879
ra_sum_io_send_wait	repagent_4	8	652118
ra_longest_io_send_wait	repagent_4	9	766
ra_io_recv	repagent_4	10	3998
ra_sum_io_recv_wait	repagent_4	11	537161
ra_longest_io_recv_wait	repagent_4	12	503

В таблице 4 приведен фрагмент вывода raw-счетчика монитора, статистика собрана за 20 минут (1200 с) работы Replication Agent. Время в столбце «value» указано в миллисекундах. Совокупное время, затраченное только на пересылку пакетов, составляет 652 с. Это довольно много. В таких случаях может помочь увеличение на машине-источнике размера TCP-буферов с помощью настраиваемых параметров net.ipv4.tcp_wmem и net.core.wmem_max.

6. Replication Server

В данном разделе описаны некоторые типовые настройки Replication Server, о которых следует знать, чтобы применять их при необходимости. Рассматривается помодульный анализ, позволяющий выявлять и устранять узкие места в работе программы. Для анализа причин снижения быстродействия часто используются счетчики Replication Server, и потому мы кратко опишем порядок съема значений счетчиков и их интерпретации.

6.1. Контроль быстродействия с помощью счетчиков

Replication Server имеет несколько десятков различных счетчиков, которые позволяют контролировать эффективность работы процесса репликации в разных точках и на разных участках. По умолчанию большинство счетчиков отключены; они включаются пользователем по мере необходимости. Для просмотра описания и состояния счетчиков используется хранимая процедура rs_helpcounter.

Снимать значения счетчиков можно несколькими способами. Обычно их собирают с помощью команды admin stats с опцией save, которая задает сбор показаний определенное число раз в течение определенного периода и сохраняет полученные сведения в RSSD. К примеру, чтобы собрать в RSSD статистику за 1 час с двадцатисекундными интервалами, следует ввести:

admin stats, “all”, save, 20, “01:00:00”

Хранимая процедура rs_dump_stats помещает показания счетчиков из соответствующих таблиц RSSD в CSV-файл, который можно проанализировать в табличном процессоре. В таблице 5 приведен фрагмент вывода процедуры rs_dump_stats. (Комментарии справа не являются частью вывода.)

Таблица 5

...	
CM: Outbound non-database connection requests	*Counter external name*
CMOBNonDBReq	*Counter display name*
13004 , , 13, -1	*Counter ID, instance ID, instance value*
Nov 5 2005 12:29:18:930AM, 103, 103, 1, 1	*Dump timestamp, observed,
Nov 5 2005 12:30:28:746AM, 103, 103, 1, 1	*total, last, max values for
Nov 5 2005 12:31:38:816AM, 107, 107, 1, 1	*the counter*
Nov 5 2005 12:32:49:416AM, 104, 104, 1, 1	
Nov 5 2005 12:33:58:766AM, 114, 114, 1, 1	
...	
Nov 5 2005 12:46:51:350AM, 107, 107, 1, 1	
ENDOFDATA	*EOD for counter*
...	

6.2. SMP Enable

В RS 15.5 параметр smp_enable по умолчанию имеет значение «ON» (включено). В этом случае Replication Server, благодаря своей многопоточной архитектуре, может эффективно использовать несколько процессорных ядер при работе на многопроцессорной системе симметричной архитектуры.

configure replication server set smp_enable to ‘on’/’off’

go

В испытаниях, проведенных на 16-процессорной машине (процессоры Intel® Xeon® X7350 @ 2,93 ГГц) при включенном параметре smp_enable в тесте на выполнение команд вставки как таковой был отмечен значительный прирост пропускной способности.

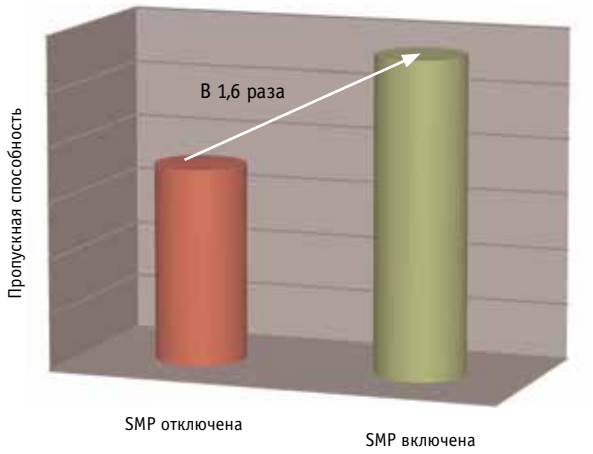

Рис. 5

В таблице 6 приведен вывод команды «top» как с включенным, так и с выключенным параметром smp_enable для тестовых прогонов. Из нее виден эффект использования сервером Replication Server дополнительных ядер при включенном параметре.

Таблица 6

Команда «TOP» выполнена несколько раз с разными интервалами. Каждая строка вывода отражает загрузку сервера в соответствующий период.										
SMP: OFF										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
17065	sybase	17	0	8921m	4.5g	9304	S	99	14.4	1:29.57	repserver
SMP: ON (It is evident from the below data that RS with up to ~488% CPU utilization is using 5 cores)										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
18708	sybase	15	0	2152m	1.3g	9280	S	488	4.2	16:10.78	repserver

6.3. Модуль System Table Services (STS)

Модуль System Table Services (STS) является интерфейсом между Replication Agent и базой данных RSSD. Он отвечает за пересылку в RSSD всех SQL-запросов. Быстродействие этого модуля можно улучшить путем локального кэширования данных RSSD — в результате может быть ускорена внутренняя обработка, выполняемая Replication Server.

В RS 15.5 к списку объектов STS, полностью кэшируемых по умолчанию, добавлены три новых объекта: rs_objects, rs_columns и rs_functions. Для сокращения накладных расходов на внутреннюю обработку полностью кэшируемых таблиц реализованы дополнительные усовершенствования.

Если значение счетчика STSCacheExceeds монитора STS велико, либо в журнале ошибок появилось подобное нижеприведенному сообщение, это говорит о том, что значение конфигурационного параметра sts_cachesize следует увеличить. Счетчик STSCacheExceeds увеличивается всякий раз, как возникает переполнение кэша, приводящее к замещению какой-либо из имеющихся записей.

I. 2010/01/27 17:32:48. A cached row for system table 'rs_config' was swapped out of the cache in order to accommodate another row. If this message appears multiple times it may be advisable to increase sts_cachesize from its present value of '5'.

Таблица 7

STSCacheExceed
11008, , 11, -1
Jan 27 2010 05:33:08:696PM, 0, 0, 0, 0
Jan 27 2010 05:33:38:845PM, 365, 365, 1, 1

Быстродействие модуля STS можно еще увеличить, выявив в RSSD интенсивно используемые таблицы и настроив для них полное кэширование, так чтобы данные этих таблиц были всегда доступны Replication Server без необходимости запрашивать информацию из RSSD. Например, если оказывается, что таблица rs_columns активно используется, ее полное кэширование в STS можно включить следующей командой:

configure replication server set sts_full_cache_rs_columns to 'on'

go

6.4. Анализ быстродействия модуля EXEC

В данном разделе описаны точки обработки данных, которым следует уделить внимание в модуле EXEC в случае, если задержки возникают в нем.

6.4.1. Отдельный поток NRM

В RS 15.5 реализован новый поток под названием NRM, который берет на себя часть работы потока EXEC. Назначение нового потока состоит в том, чтобы сократить время возврата к Replication Agent и тем самым как повысить его быстродействие, так и увеличить общую скорость приема данных. В более ранних выпусках RS поток EXEC используется для разбора, нормализации и упаковки поступающих пакетов команд LTL и постановки их в очередь с целью стабилизации очередей до отправки ответа Replication Agent. В выпуске 15.5 он только разбирает команды, после чего передает их потоку NRM для дальнейшей обработки. Благодаря этому ответ Replication Agent отправляется гораздо быстрее.

На рисунке 6 показано, как наличие потока NRM отражается на быстродействии Replication Agent.

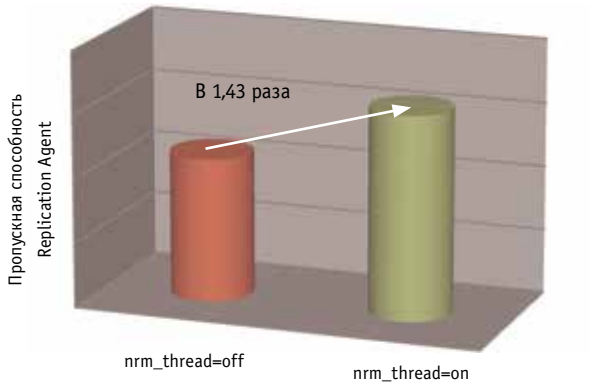

Рис. 6

6.4.2. Время ожидания завершения операций записи

Счетчик RAWritesWaitTime раздела Replication Agent в статистике Replication Server отражает время, в течение которого модулю EXEC приходилось ждать завершения операций записи.

Если этот счетчик принимает большое значение лишь изредка, ситуацию можно улучшить, увеличив значение конфигурационного параметра exec_sqm_write_request_limit. Указанный параметр задает максимальное количество оставшихся байтов, которые EXEC может удерживать до начала ожидания записи части блока данных во входную очередь. Значение по умолчанию — 1M. Это означает, что поток EXEC переходит в состояние блокировки лишь в том случае, когда размер данных, ожидающих записи, начинает превышать 1 Мбайт.

Если же значение счетчика RAWritesWaitTime постоянно держится на высоком уровне, значит, скорость записывающего модуля SQM меньше скорости, с которой модуль EXEC обрабатывает данные, так что поток SQM Write за ним не успевает. В параграфе 6.6.2 предложены некоторые способы ускорения записи данных в целях стабилизации очередей.

6.4.3. Исполнение команды «get truncation»

Всякий раз после отправки числа команд, равного scan batch size, Replication Agent запрашивает у Replication Server последнюю вторичную точку усечения (scan batch size является конфигурационным параметром Replication Agent). Для этого он отправляет специальный пакет, содержащий команду «get truncation». Получая этот пакет, поток EXEC в Replication Server выполняет эту команду сам (в отличие от других команд LTL, которые он передает потоку NRM). EXEC инициирует в RSSD транзакцию под названием Save_Locator, чтобы обновить последний записанный OQID (Origin Queue Identifier) в таблице rs_locator и возвращает ту же информацию Replication Agent.

Отмечено, что при использовании внедренной RSSD выполнение транзакции Save_Locator сопровождается относительно высокими накладными расходами. В этом случае может оказаться полезным в настройке Replication Agent увеличить параметр scan batch size. С точки зрения быстродействия, чем больше размер этого пакета, тем лучше, поскольку сокращается число коммуникаций с Replication Server и, соответственно, число выполнения транзакций Save_Locator. Однако при этом может задерживаться пересылка вторичной точки усечения в сервере-источнике, что, в свою очередь, задерживает усечение журнала.

Как правило, частота обновления OQID порядка раза в минуту является приемлемым значением — с одной стороны, транзакции Save_Locator в RSSD выполняются не слишком часто, с другой, вторичная точка усечения основной системы ASE передается достаточно часто, чтобы ASE могла усечь журнал до того, как он переполнится. Для контроля числа обновлений OQID в RSSD по инициативе модуля EXEC можно использовать счетчик UpdsRslocater агента репликации.

На рис. 7 показан прирост пропускной способности, достигнутый в одном из тестов путем увеличения параметра scan batch size с 1000 до 20000.

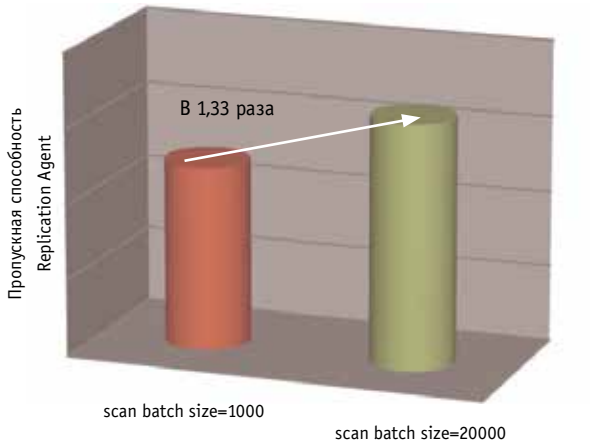

Рис. 7

6.5. Анализ быстродействия модуля DIST

6.5.1. Недостаточно большой размер кэша SQT

Если счетчик TransRemoved имеет ненулевое значение или в журнале ошибок Replication Server появилось сообщение, подобное приведенному ниже, это может быть признаком того, что размер кэша SQT слишком мал для текущих потребностей.

```
2009/10/28 23:49:13. WARNING #24068 SQT(102:1 DIST PRIM.tpcc) -
t/sqtint.c(1370)
SQT cache size is too low to load more than one transaction into the cache.
```

Таблица 8

SQT: Transactions removed from memory
TransRemoved
Jan 25 2010 12:35:08:362PM, 0, 0, 0, 0
Jan 25 2010 12:35:23:471PM, 8, 8, 1, 1
Jan 25 2010 12:35:38:811PM, 12, 12, 1, 1
Jan 25 2010 12:35:53:860PM, 17, 17, 1, 1

Ненулевые значения выше отражают количество транзакций, составляющие команды которых были удалены из кэша SQT. Удаление транзакций могло быть вызвано одной транзакцией, превысившей доступную емкость кэша либо содержащейся в кэше объемной транзакцией, сведения о фиксации которой еще не поступили. Модуль SQT, при загрузке кэша, если кэш SQT полон и нет закрытой транзакции (закрытой транзакцией называется транзакция, фиксация которой уже отмечена) либо прочитанной транзакции (то есть транзакции, содержание которой считано DIST/DSI), ищет в списке открытых транзакций самую большую и высвобождает занятую ею память. Это снижает быстродействие, поскольку в дальнейшем эту транзакцию потребуется считать с диска. Чтобы попытаться исключить такую ситуацию, можно увеличить размер кэша SQT. Необходимо, однако, учитывать при этом, что если транзакции чересчур велики (достигая размера в миллионы команд), то даже самый большой кэш окажется бесполезным.

6.6. Анализ быстродействия модуля SQM

6.6.1. Настройка размера блока очереди

Начиная с Replication Server 15.5 пользователь может увеличить размер блока очереди, чтобы в одном блоке уместилось больше данных. В ранних выпусках блок имел фиксированный размер 16 Кбайт. В выпуске 15.5 его можно увеличить вплоть до 256 Кбайт. Увеличение размера блока не только ускоряет ввод-вывод, но также и уменьшает время обработки благодаря уменьшению количества выделений и высвобождений сегментов памяти. При этом, в случае если узким местом является модуль SQM, должна значительно повыситься скорость репликации.

configure replication server set block_size to '<size>' with shutdown

go

Как видно из рисунка 8, в проведенном тесте сквозная пропускная способность системы репликации благодаря увеличению размера блока с 16 до 128 Кбайт возросла на 15%.

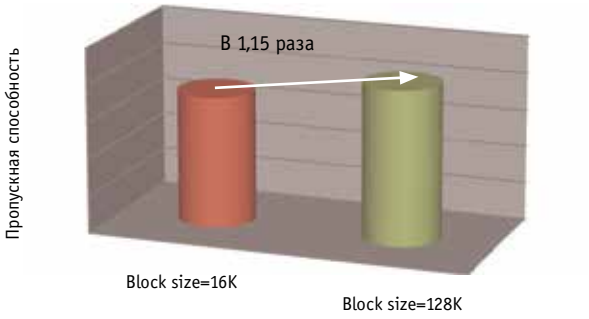

Рис. 8

6.6.2. Достаточно ли быстро выполняется запись?

Счетчик Replication Server SQMWriteTime отражает время, затраченное на запись данных постоянных очередей. Поскольку эта запись выполняется отдельным потоком (потоком SQM), то значение данного процесса для скорости обработки зависит от количества времени, в течение которого клиентам приходится ожидать завершения записи. Признаком проблемы, например, может служить на стороне-получателе, где клиентом является модуль EXEC, большое значение счетчика EXEC RAWriteWaitsTime, а на стороне-отправителе, где клиентом является модуль DIST — счетчика DIST DISTTDDeliverTime.

Если оказалось, что проблема заключается в скорости записи, то помимо увеличения размера блока, описанного в предыдущем параграфе, можно настроить еще два параметра Replication Server — sqm_write_flush и sqm_page_size.

Конфигурационный параметр sqm_write_flush применим лишь в том случае, если постоянные очереди размещены в файловой системе. Он определяет, следует ли выгружать на диск данные, записываемые в буферы памяти, до того как операция записи будет завершена. По умолчанию этот параметр имеет значение «on», то есть данные выгружаются на диск до завершения записи.

Скорость записи можно увеличить, присвоив параметру sqm_write_flush значение «dio». При этом Replication Server станет использовать для записи и чтения постоянных очередей прямой ввод-вывод (direct IO), минуя буферы файловой системы и тем самым исключая издержки на манипулирование ими.

Параметр sqm_page_size задает размер страницы постоянной очереди в блоках размером block_size. Например, если установить sqm_page_size в 16, то Replication Server будет записывать в постоянную очередь фрагменты размером 16*block_size. При увеличении параметра за один вызов функции записи в постоянные очереди будет записываться больше данных, и быстродействие значительно возрастет. В диаграмме на рис. 8 приведены значения счетчика SQMWriteTime для тестового прогона с постоянными очередями в файловой системе.

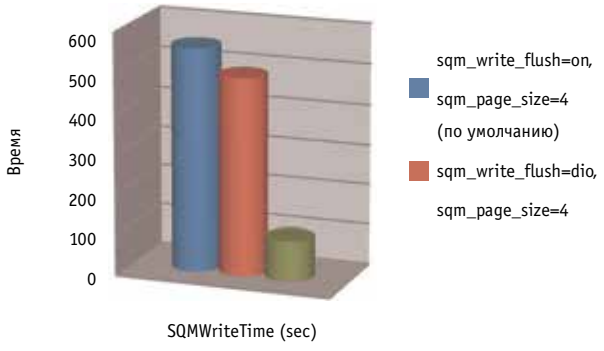

Рис. 9

6.6.2.1. Достаточно ли велики постоянные очереди?

Иногда бывает, что клиенты потока SQM Writer должны ждать слишком долго, даже если фактически запись выполняется быстро. Например, значение счетчика RAWriteWaitsTime может быть чересчур велико, при этом счетчик SQMWriteTime для соответствующей постоянной очереди может иметь не столь большое значение. Такое случается, когда постоянная очередь заполнена и все последующие операции записи вынуждены ожидать освобождения какого-либо сегмента, так что запись замедляется. О присутствии этой проблемы сигнализирует сообщение, подобное следующему:

W. 2009/06/09 04:25:16. WARNING #6047 dSEG( ) – qm/sqmsp.c(896)

SQM_ADD_SEGMENT ('103:0'): Going to wait for a segment to be freed. Administrative action is needed

О проблеме может свидетельствовать и счетчик SQMWaitSegTime, отражающий количество времени, в течение которого записывающий поток SQM должен ожидать выделения сегмента.

Таблица 9

SQM: Time waiting for segment allocation
SQMWaitSegTime
6059 , , 102, 1
Jun 9 2009 3:55:21:086AM, 1, 14, 14, 14
...
Jun 9 2009 4:00:53:633AM, 63, 24524, 209, 691 Jun 9 2009 4:01:23:726AM, 64, 26207, 409, 648 Jun 9 2009 4:01:53:930AM, 72, 26639, 447, 534 Jun 9 2009 4:02:24:236AM, 63, 26636, 423, 710

В таких случаях бывает целесообразно увеличить размер постоянных очередей.

6.6.3. Достаточно ли быстро выполняется чтение?

6.6.3.1. Удовлетворяет ли кэш SQM потребности в чтении?

Счетчик SQMRReadTime отражает время, затрачиваемое модулем SQMR на чтение блоков из постоянных очередей. Подчеркнем, что в отличие от записывающего потока SQM, SQMR является не отдельным потоком, а частью самого потока-клиента (SQT для получателя и DSI-S для отправителя). Большое значение SQMRReadTime свидетельствует о том, что вызывающий поток затрачивает много времени на считывание блоков.

Таблица 10

SQMRReadTime 62011, , 102, 11 Apr 01 2010 02:53:15:412AM, 0, 0, 0, 0 Apr 01 2010 02:53:45:425AM, 6050, 4197, 0, 51 Apr 01 2010 02:54:15:738AM, 15688, 11866, 0, 54 Apr 01 2010 02:54:45:786AM, 16281, 12061, 0, 68 Apr 01 2010 02:55:15:839AM, 20105, 16434, 1, 54 ENDOFDATA	
SQMR: Blocks read from queue BlocksRead 62002, , 102, 11 Apr 01 2010 02:53:15:412AM, 0, 0, 0, 0 Apr 01 2010 02:53:45:425AM, 15372, 15372, 1, 1 ENDOFDATA	SQMR: Blocks read from cache BlocksReadCached 62004, , 102, 11 Apr 01 2010 02:53:15:412AM, 0, 0, 0, 0 Apr 01 2010 02:53:45:425AM, 9280, 9280, 1, 1 ENDOFDATA

В идеальной ситуации все чтение должно производиться из кэша и потребности в обращении к диску возникать не должно. Часто, однако, этого не происходит. Проконтролировать эффективность кэша SQM можно, сравнив количество блоков, считанных из кэша (счетчик BlocksReadCached), с общим количеством прочитанных блоков (BlocksRead).

Если все чтение производится из кэша SQM, значения обоих счетчиков будут точно совпадать. Если же окажется, что кэш удовлетворяет лишь небольшое число запросов на чтение, возможно, стоит увеличить размер кэша SQM с помощью конфигурационного параметра sqm cache size.

Заметим, что обеспечить удовлетворение всех запросов на чтение из кэша возможно не всегда. Если клиент модуля SQMR не в состоянии обрабатывать данные быстрее, чем их выдает клиент соответствующего записывающего потока SQM, то содержимое кэша SQM будет переписываться новыми данными прежде, чем старые блоки будут считаны, и при последующих операциях чтения потребуется обращение к диску. Увеличение кэша SQM полезно, когда вероятно значительное увеличение объемов данных.

6.7. Анализ быстродействия модуля DSI

6.7.1. Высокообъемная адаптивная репликация (dsi_compile_enable)

Обычный процесс репликации непрерывен, то есть реплицируется каждое зарегистрированное в журнале изменение записи в БД. HVAR (High Volume Adaptive Replication — высокообъемная адаптивная репликация) — это новый способ репликации, когда в базу-приемник передаются только итоговые изменения записей. Данные при этом обрабатываются по следующей схеме:

1. Множество транзакций группируется в одну.
2. Механизм компиляции HVAR находит изменения по отношению к одним и тем же записям, в порядке следования согласно журналу, и преобразует их в итоговые изменения. Последние сохраняются в базе итоговых изменений в трех таблицах: вставки, изменения и удаления. Во всех таблицах используются первичные ключи, заданные репликационными определениями. Для таблиц, где таких определений нет, в качестве первичных ключей используются все столбцы.
3. Скомпилированные транзакции массово применяются к БД-приемнику. Итоговые изменения перед применением сортируются по таблицам-репликам и по операциям (вставка, изменение, удаление). Поскольку ни IQ, ни ASE не поддерживают массового изменения и удаления, механизм HVAR массово вставляет итоговые изменения в рабочие таблицы в БД-реплике, после чего к таблицам-репликам применяются операции JOIN-UPDATE или JOIN-DELETE.

Данная возможность позволяет значительно повысить пропускную способность при репликации журнала транзакций OLTP. Диаграмма на рис. 10 отражает распределение всех переданных команд по размерам пакетов массовых операций в лабораторном эксперименте.

Как видно из диаграммы, более 50% команд были переданы группами по 1000-5000 команд. Подчеркнем, что это группы команд вставки, либо обновления, либо удаления для конкретной таблицы (а не просто различные команды для разных таблиц, объединенные в пакет, как было до реализации HVAR), передаваемые за один прием с помощью API массовой загрузки. Такая передача значительно эффективнее, чем раздельная передача операций языковыми командами, и именно благодаря ей достигается повышение быстродействия при использовании HVAR.

Рис. 10

Максимальное число команд, которые механизм HVAR может объединить в одну транзакцию, управляется конфигурационным параметром dsi_compile_max_cmds. Однако фактическое число группируемых команд может быть намного меньше, в случае если кэш SQT недостаточно велик. Чтобы HVAR мог группировать большое число транзакций, кэш должен быть достаточного размера.

6.7.2. Параллельные потоки DSI

Если в мониторе DSI отмечено большое значение счетчика AllThreadsInUse или AllLargeThreadsInUse (см. пример ниже), возможно, стоит использовать параллельные потоки DSI, задав параметры dsi_num_threads или dsi_num_large_xact_threads соответственно. Счетчик увеличивается всякий раз, когда параллельной транзакции приходится ждать из-за отсутствия доступных параллельных потоков DSI. Риск включения параллельной обработки DSI состоит в том, что между транзакциями разных подключений DSI могут возникнуть конфликты. Одновременное применение транзакций к БД-реплике может вызвать различные блокировки БД с разных сторон и конкуренцию за серверные ресурсы. Если в БД-реплике отмечается возрастание количества взаимных блокировок, рекомендуется уменьшить число потоков.

Таблица 11

AllThreadsInUse
5057, , 103, -1
Jan 21 2010 04:41:42:782PM, 49, 49, 1, 1
Jan 21 2010 04:41:57:999PM, 1015,1015, 1, 1
Jan 21 2010 04:42:13:032PM, 567, 567, 1, 1

При использовании HVAR объем внутренней обработки, выполняемой потоком DSIEXEC, естественным образом возрастает. Поэтому, как правило, целесообразно иметь два потока DSIEXEC, даже если применяется метод сериализации «wait after commit». При использовании данного метода собственно параллельная обработка в БД-реплике не выполняется, однако один поток может выполнять работу, в то время как второй отправляет транзакции в базу-реплику. В пробной репликации из ASE в IQ при включении параллельной обработки DSI с двумя потоками DSIEXEC был отмечен примерно 20-процентный прирост производительности.

6.7.3. Массовое копирование

При обычной репликации Replication Server, копируя данные в Adaptive Server, формирует SQL-команду вставки и отправляет ее Adaptive Server, после чего ожидает обработки записи и возврата результата операции. При репликации таким способом больших пакетов данных, например при пакетной обработке в конце дня или консолидации сделок, быстродействие Replication Server заметно снижается.

В версии Replication Server 15.2 в целях ускорения репликации больших пакетов команд вставки в одну и ту же таблицу в Adaptive Server® Enterprise 12.0 и более старших версий введена поддержка массового копирования. Массовое копирование реализовано в модуле DSI Replication Server на основе библиотеки Open Client™ Open Server™ Bulk-Library.

Массовые операции в DSI управляются параметрами подключения БД dsi_bulk_copy и dsi_bulk_threshold.

7. Сервер данных — реплика

7.1. ASE

7.1.1. Кэш операторов и функция literal autoparam

Известно, что когда репликация осуществляется с помощью языковых команд (в отличие от dsi_compile_cmds или функции dsi_bulk_copy, использующих интерфейс массовой загрузки, либо функции dynamic_sql, которая использует динамический SQL), использование такой возможности ASE, как кэш операторов, дает значительный выигрыш.

В кэше операторов ASE хранит текст произвольных операторов SQL. Система сравнивает вновь полученные SQL-операторы с имеющимися в кэше и, если находит совпадение, использует план, сохраненный в кэше в момент первоначального исполнения оператора. Таким образом, отпадает необходимость повторной компиляции SQL-операторов, и затраты на компиляцию запросов при повторном выполнении одних и тех же операторов уменьшаются.

Для того чтобы добиться максимального прироста быстродействия, эту возможность следует использовать вместе с режимомliteral autoparam, введенным в ASE 15.0.1. В ранних версиях ASE при обработке двух запросов, идентичных друг другу во всем, кроме одного или нескольких литералов, в кэш операторов помещалось два отдельных плана запросов. Начиная с ASE 15.0.1 значения литералов в SQL-запросах могут автоматически конвертироваться в описания параметров (подобные переменным). Данный режим включается командой «enableliteral autoparam» опции sp_configure. Настроить кэш операторов, а также включить режимliteral autoparam можно, выполнив хранимую процедуру sp_configure:

sp_configure “statement cache size”, 20000

go

sp_configure “enable literal autoparam”, 1

go

Настройка кэша операторов в базе-реплике ASE существенно ускоряет репликацию. Эффективность кэша можно проверить с помощью отчета sysmon сервера-реплики (см. табл. 12).

Таблица 12

Procedure Cache Management	per sec	per xact	count	% of total
SQL Statement Cache:				
Statements Cached	0.1	0.0	23	n/a
Statements Found in Cache	22.7	0.4	6946	n/a
Statements Not Found	0.1	0.0	23	n/a
Statements Dropped	0.0	0.0	0	n/a
Statements Restored	8.8	0.2	2689	n/a
Statements Not Cached	0.0	0.0	0	n/a

Из таблицы следует, что в кэше было обнаружено 6946 операторов, и всего 23 операторов там не нашлось. Таким образом, большая доля операторов была взята из кэша, из чего видно, что размер его достаточен.

7.2. IQ

7.2.1. Влияние индексов в таблицах IQ

Установлено, что наличие иных индексов в таблицах IQ, нежели индекс FP по умолчанию, негативно сказывается на скорости репликации. Причиной может быть задержка при обновлении этих индексов, которое производится при каждой DML-операции. Указанные индексы, однако, могут требоваться для регулярно выполняемых на сервере IQ задач, таких как генерация отчетов. Следует тщательно рассчитать фактический выигрыш от них и выяснить целесообразность их использования, учитывая их влияние на процесс репликации в IQ. Отмечено, что на скорость исполнения алгоритмов соединения, используемых HVAR в операциях соединения с рабочими таблицами для репликации удалений и изменений, никакие индексы в таблицах IQ положительного влияния не оказывают.

В таблицах IQ наиболее широко применяются уникальные индексы HG для столбцов — первичных ключей (они подобны кластеризованным индексам в ASE). Рис. 11 позволяет сравнить быстродействие репликации в тесте OLTP при наличии таких индексов в таблицах-репликах в IQ.

Влияние HG-индексов первичных ключей в таблицах-репликах на скорость репликации

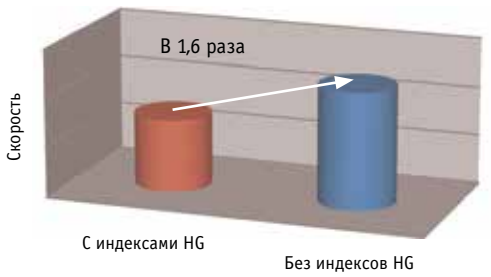

Рис. 11

7.2.2. Ускорение операций изменения в IQ с помощью u2di

В случае, если репликация в IQ сопровождается изменениями в таблице каких-либо символьных столбцов большой длины (тысячи байт), для обработки этой таблицы может быть полезно присвоить опции Replication Server dsi_command_convert значение «u2di» для конкретного подключения. При этом операции изменения будут замещаться последовательностью команд удаление — вставка (update to delete/insert). Подчеркнем, что опция dsi_command_convert может применяться к определенной таблице — нет необходимости задавать ее для подключения целиком. Обратной стороной использования u2di в таблицах IQ является некоторая фрагментация таблицы.

Данная подстройка, возможно, не будет требоваться в будущих выпусках Replication Server (начиная с RS 15.5 EDS#2) — работа над оптимизацией репликации изменений уже ведется.

В проведенном тесте репликации OLTP в одной из таблиц изменялись два символьных столбца длиной 250 байт каждый. Рис. 12 позволяет сравнить быстродействие в режиме u2di и в обычном.

Ускорение репликации в режиме u2di

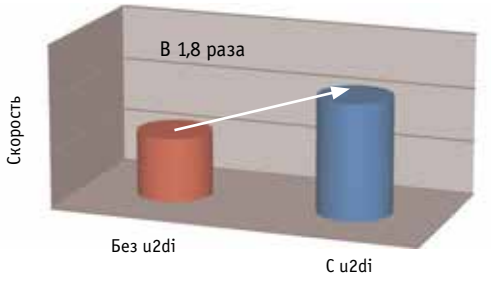

Рис. 12

8. Заключение

Как с очевидностью свидетельствуют проведенные тесты, RS 15.5 обеспечивает превосходное быстродействие как при использовании в качестве приемника ASE, так и IQ. Для OLTP-нагрузок быстродействие по сравнению с RS 15.2 — более чем двукратное в случае репликации из ASE в ASE; достигается также выигрыш вплоть до шестикратного для некоторых пользовательских приложений с профилями транзакций, ориентированными на вставку. При использовании RS 15.5 становится возможной репликация в IQ в реальном времени.

Приведенные в настоящем руководстве рекомендации по настройке можно применять тогда, когда в соответствующих точках обработки данных возникают задержки. Снимаемая со счетчиков монитора информация показывает, на какие моменты обработки данных следует обратить внимание при анализе и оптимизации быстродействия системы репликации.

Sybase, Inc.
Worldwide Headquarters
One Sybase Drive
Dublin, CA 94568-7902
U.S.A
1 800 8 sybase

www.sybase.com

Sybase CIS
115114, Москва,
Дербеневская набережная,
д. 7, стр. 16
+7 (495) 797-4774

www.sybase.ru

SYBASE®
An SAP Company