УДК 004.67

ГУСЕВ Е.И.

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЁННОЙ КЛАСТЕРНОЙ СИСТЕМЫ ИСПОЛЬЗУЮЩЕЙ SHARED EVERYTHING ПОДХОД (ORACLE RAC)

Предложена математическая модель функционирования распределённой общей памяти на примере Global Cache Fusion (GCF). GCF – реализация Shared Everything подхода в Oracle RAC. Выделены основные характеристики трафика для работы с моделью, а также определена целевая функция для анализа эффективности алгоритмов оптимизации доступа к странице памяти в распределённой системе.

There are proposed a mathematical model of functioning distributed shared memory by the example of the Global Cache Fusion (GCF). GCF is a realization of Shared Everything approach implemented in Oracle RAC. Discovered fundamental characteristics of traffic for using the model, and also defined the objective function for analysis of effectiveness of access to memory page optimization algorithms in a distributed system.

Постановка задачи

Ключевыми задачами системы с точки зрения производительности является увеличение пропускной способности системы и уменьшение времени отклика. В работе [1] ключевым параметром, характеризующим пропускную способность многопользовательской СУБД является SESSTIME (ST) – суммарное время всех задержек во всех сессиях, кроме задержек действий клиента. Приведём формулу:

$$ST = \int_0^t \left(\nu(\tau_{prc})\right) dt = \int_0^t \left(\nu(kt_{sb})\right) dt$$

$$ST = \int_0^t \nu\left(k\left(t_{acc} + t_{prc}\right)\right) dt$$

где τ_{prc} – время обработки одного запроса

t_{sb} – время обработки одного блока

ν – интенсивность запросов к базе

k – количество обрабатываемых в одном запросе блоков

t_{acc} – время доступа к блоку

t_{prc} – время выполнения операций над блоком

Поскольку одной из ключевых задач системы с точки зрения производительности является уменьшение ST при неизменной интенсивности запросов, исследование возможности уменьшения t_{acc} и t_{prc} являются приоритетными задачами. В данной статье мы построим временную модель доступа к блоку в распределённой кластерной системе использующей shared everything подход на примере Oracle RAC. Также заметим, что уменьшение среднего времени доступа к блоку влияет не только на уменьшение ST, но и на время отклика, что представляет интерес в контексте исследования устойчивости на коллапс по блокировкам системы [1].

Важно выделить среди факторов, влияющих на среднее время отклика, базовые характеристики системы, базовые характеристики трафика и производные характеристики. Базовыми характеристиками системы будем называть факторы, зависящие ТОЛЬКО от архитектуры системы и физических возможностей, например, скорость передачи данных между нодами, количество нод или размер кэша на каждой ноде. Базовыми характеристиками трафика назовём характеристики зависящие ТОЛЬКО от набора интенсивностей классифицированных как по типу запроса так и по тому к какому ресурсу запрос. Например, общая интенсивность запросов, соотношение запросов чтения и записи, вероятность конфликта и т. д. И производными параметрами назовём характеристики, которые обусловлены взаимодействием трафика с системой. Например, вероятности состояний блока в кэше, вероятность попадания в кэш, среднее время блокирования и т.д.

1. Описание принципов алгоритма доступа к блоку в Oracle RAC

Следует сказать, что в реальных кластерных системах для взаимосвязи узлов кластера 2PC [4] и 2PL[5] протоколы не используются. Это связано с резким снижением производительности протоколов при увеличении интенсивности конфликта за данные [1,2]. Ключевой особенностью Oracle RAC является Shared Everything подход [3]. Это предполагает как общий диск, так и общую оперативную память на всех нодах. Однако, сразу оговоримся, что «общая память» – в данном случае – условное обозначение. На практике, каждая нода имеет собственную оперативную память, но

осуществляется пересылка блока (страницы) данных от одной ноды к другой по запросу, если блок закеширован на другой ноде [6]. При такой организации возникает задача отслеживания состояния блоков на всех нодах: а именно: какой блок на какой ноде закеширован, и на какой момент времени. Задача решается введением ноды владельца (мастер-ноды) для каждого блока. Все ноды будут слать запросы на изменение состояния блока к этой мастер ноде. Состояние блока может быть XCUR (eXclusive CURrent – эксклюзивное текущее), SCUR (Shared CURrent – разделяемое текущее), CR (Consitent Read – целостное чтение) и PI (Past Image exist – после изменений, удобен при восстановлении, чаще всего следующим действием сконвертируется в CR). Поясним состояния: если мы собираемся изменять блок – мы запрашиваем у мастер ноды блокировку на блок XCUR. Если собираемся его читать – то SCUR. Мастер нода отслеживает каким нодам выданы SCUR блокировки, а какой XCUR. Не более одной копии любого блока закэшированного в XCUR состоянии может быть в каждый момент времени. Для модификации блока, процесс должен получить на него состояние XCUR. CR предполагает, что блок актуален на некоторый момент времени в прошлом, но на данный момент существует более поздняя версия блока. PI – состояние в котором оказывается блок после состояние XCUR, т.е. другая нода запросила XCUR либо SCUR блокировку, и нода, ранее удерживающая XCUR, её отдала. Блок получается в состоянии PI, которое впоследствии может быть сконвертировано в CR. Важно сказать, что один и тот же блок может одновременно находиться на разных нодах в своём состоянии для каждой ноды. С точки зрения исследования, важно выделить 5 состояний блока, с учётом наличия его различных версий и состояний каждой версии на каждой ноде в глобальном кеше:

1. XCUR+n*CR: n – целое, неотрицательное число. В этом состоянии на одной ноде на одной из версий блока XCUR блокировка, а на остальных нодах – версии блока в CR. Количество CR версий блока на каждой ноде произвольно, столько, сколько версий блока понадобилось для работы ноде, причем, версии в CR состояниях могут быть и на ноде, где удерживается XCUR блокировка. Обозначим состояние xcur.
2. k*SCUR+n*CR: k – натуральное число, n – целое, неотрицательное число. На каждой ноде не более одной версии с SCUR блокировкой, но хотя бы одна версия блока в состоянии SCUR существует на какой-либо из нод. Одновременно с этим, в CR состояниях на всех нодах, может быть неограниченное количество версий блока. Обозначим состояние scur.
3. n*CR (XCUR или SCUR вылетел): В глобальном кэше не осталось XCUR или SCUR версий блока. Стоит заметить, что при первого обращения к блоку SCUR или XCUR запросом блок будет зачитан с диска для обработки, после которой останется в состоянии SCUR либо XCUR в зависимости от запроса. Однако, при запросе CR состояния – необходимости зачитывать с диска блок нет, если блок на нужный момент времени остался закэшированным. В данной статье рассмотрен классический GCF алгоритм, позволяющий повторно не собирать блок, а использовать ЛОКАЛЬНО закешированную версию в дальнейшем. Помимо классического подхода в качестве техник оптимизации может быть предложена техника поиска CR блока в ГЛОБАЛЬНОМ кэше. Если мастер нода сохраняет историю получения XCUR блокировок и историю CR запросов к ней она может указать на какой ноде существуют уже собранные на нужный момент CR блоки. Но, как было сказано выше, в этой статье рассматривается классический подход. Обозначим это состояние «cr only».
4. Блока нет в глобальном кэше. Обозначим состояние «нет».
5. Мастер нода при обращении к другой ноде обнаруживает, что ранее закэшированная в состоянии SCUR или XCUR версия блока вылетела. Обозначим состояние «вылетел».

2. Сценарии алгоритма доступа к блоку в Oracle RAC

Теперь перейдём к описанию последовательностей действий (сценариев), в контексте получения запроса блокировок. Стоит сразу отметить, что сценарии разные в случаях, когда не существует закешированного в глобальном кэше блока, существует один или несколько закешированных блоков с SCUR блокировкой и уже существует закешированный блок с XCUR блоки-ровкой на какой-либо из нод. Таких сценариев 9. Распишем эти сценарии:

Сценарий 1. В случае если блок незакеширован ни на одной из нод (о том что блок незакеширован мастер ноде известно) имеем следующий сценарий:

1. Запрос XCUR/SCUR блокировки у мастер-ноды

2. Ответ мастер ноды об отсутствии блока в глобальном кэше.

3. Чтение блока с общего диска.

Обозначим его как чтение с диска (read from disk, rd1). Распишем временную модель для сценария:

$$t_{rd1} = 2t_{net} + t_{io} \ (1)$$

где t_{io} – время чтения блока с диска

t_{net} – среднее время пересылки служебного сетевого пакета между нодами

Сценарий 2. Если же мастер нода имеет информацию о закешированном блоке, но при обращении оказывается, что блок вылетел из кэша, то сценарий такой:

1. Запрос XCUR/SCUR блокировки у мастер-ноды

2. Отправка ноде, где закеширован блок инструкции, переслать блок запросившей ноде.

3. Ответ ноды, где блок вылетел из кэша, мастер-ноде об отсутствии блока в кэше.

4. Ответ мастер ноды запросившей ноде об отсутствии блока в глобальном кэше.

5. Чтение блока с общего диска.

Обозначим его как чтение с диска (read from disk, rd2)

$$t_{rd2} = 4t_{net} + t_{io} \ (2)$$

Сценарий 3. Если же мастер нода имеет информацию о закешированном блоке и блок в глобальном кэше в состоянии SCUR, имеем:

1. Запрос XCUR блокировки у мастер-ноды

2. Мастер-нода отсылает всем нодам у которых этот блок в состоянии SCUR сообщение, что поступил запрос на изменение блока (XCUR) и с какой ноды.

3. Состояние блока на нодах где был SCUR, меняется на CR.

4. Одна из нод пересылает ноде, запросившей XCUR сам блок.

5. Нода, захватившая XCUR блокировку, подтверждает мастер ноде её захват.

Обозначим сценарий как запись-чтение (write-read, wr) и распишем временную модель. Поскольку начать обработку можно параллельно с последним шагом, имеем:

$$t_{wr} = 2t_{net} + t_{send} \ (3)$$

где t_{send} – время пересылки, включая время записи в память страницы от одной ноды к другой.

Сценарий 4. Если же мастер нода имеет информацию о закешированном блоке и блок в глобальном кэше в состоянии XCUR, имеем:

1. Запрос XCUR блокировки у мастер-ноды

2. Мастер-нода отсылает ноде, у которой этот блок в состоянии XCUR, сообщение, что поступил запрос на изменение блока (XCUR) и с какой ноды.

3. Ожидание завершения обработки

4. Состояние блока на ноде, где был XCUR, меняется на PI.

5. Нода, где был XCUR, пересылает ноде, запросившей XCUR, сам блок.

6. Нода, захватившая XCUR блокировку, подтверждает мастер ноде её захват.

Обозначим сценарий как запись-чтение (write-write, ww), как и в случае выше, обработка начинается одновременно с последним шагом. Имеем:

$$t_{ww} = 2t_{net} + t_{send} + t_{fp} \ (4)$$

где t_{fp} – время ожидания завершения обработки, из-за конфликта за блок

Аналогично распишем 2 сценария получения SCUR блокировки при наличии блока в глобальном кэше. Различие между ними в том, закешированы ли версии блоков с SCUR или XCUR блокировками.

Сценарий 5. Распишем случай, когда на одной или нескольких нодах закешированы версии блока с блокировкой SCUR:

1. Запрос SCUR блокировки у мастер ноды.

2. Пересылка мастер нодой ноде, на которой закэширован блок с SCUR блокировкой, инструкции переслать текущую версию блока запросившей ноде.

3. Пересылка блока с SCUR блокировкой запросившей ноде.

4. Пересылка подтверждения получения блокировки мастер ноде.

Обозначим этот сценарий как чтение-чтение (read-read, rr), также обработка начинается одновременно с последним шагом:

$$t_{rr} = 2t_{net} + t_{send} \ (5)$$

Сценарий 5. Если существует версия блока с XCUR блокировкой на какой либо из нод, то последовательность действий следующая:

1. Запрос SCUR блокировки у мастер ноды.

2. Пересылка мастер нодой ноде, на которой закэширован блок с XCUR блокировкой, инструкции переслать текущую версию блока запросившей ноде.

3. Ожидание завершения обработки

4. Пересылка блока запросившей ноде. При этом XCUR блокировка может быть конвертирована (понижена) в SCUR, и тогда ноде, запросившей SCUR – SCUR и будет предоставлен или же блокировка останется

XCUR, а запросившей ноде будет отдан блок в состоянии CR – но на текущий момент времени. Решение о понижении блокировки принимается на втором шаге сценария мастер-нодой. Отметим, что Oracle RAC понижает блокировку до SCUR в случае повторного запроса SCUR блокировки. С точки зрения временных задер-жек сценария время доступа остаётся неизмен-ным и не зависит от понижения блокировки при выполнении сценария. Однако условия пониже-ния блокировки влияют на то, в каком состоя-нии окажется блок после обработки запроса. Обозначим вероятность понижения блокировки как $p_{down.}$ Заметим, что в формулах этой статьи p_{down} встречаться не будет, тем не менее, выделим эту вероятность как производный параметр. При описании базовых характеристик трафика и вычислении производных параметров алгоритм принятия решения о понижении может быть предметом оптимизации, но в данной статье этот вопрос не рассматривается.

5. Пересылка подтверждения захвата блокировки мастер ноде.

Обозначим этот сценарий как чтение-запись (read-write, rw)

$$t_{rw} = 2t_{net} + t_{send} + t_{fp} \quad (6)$$

В системе существует также сценарий запись на диск (write-disk wd). Но с точки зрения анализа времени отклика или суммарных задержек во всех сессиях, сценарий не представляет интереса, поскольку отложенная запись на диск осуществляется фоновым процессом DBWn асинхронно по мере накопления изменённых блоков в кэшах нод. Анализ представляет интерес с точки зрения исследования конфликтов, поскольку во время записи на диск выставляется XCUR блокировка. В данном исследовании, поскольку задержки фоновых процессов мы не классифицируем как часть ST, время потраченное на фоновую запись на диск в итоговую формулу задержек не войдёт.

Два самых неприятных, с точки зрения моделирования сценария – это запросы CR состояния. Проблема в самой архитектуре Oracle. В данном исследовании мы будем считать что запрос CR состояния может быть преобразован либо в запрос нескольких SCUR блоков, либо, если блок для которого затребованный момент времени совпадает с временем хранимой в кэше копии, в сценарий чтения готового блока из локального (либо глобального) кэша. Таким образом, имеем сценарии:

Сценарий 8 (оптимизированный): нужный CR блок на нужный момент времени оказался в кэше (Обозначим сценарий как crp*)

Сценарий 9: Нужного CR блока не оказалось в кэше и пришлось его собирать. (Обозначим сценарий как crp).

Рассмотрим 9й сценарий (crp): Если в локальном кэше нет требуемого блока на запрошенный момент времени, то для реконструкции блока к моменту времени в прошлом должны быть задействовованы блоки UNDO-сегментов. Система выполняет 5 шагов:

1. Получение SCUR блокировки на изменяемый блок
2. Чтение блока для определения задействованных в изменении блока транзакций.
3. Определение UNDO сегментов, задействованных незавершенными транзакциями
4. Получение SCUR блокировок на блоки этих UNDO сегментов
5. Применение UNDO информации для восстановления блока на требуемый момент времени.

Фактически запрашивается состояние SCUR как на требуемый блок так и еще на один или несколько блоков UNDO – т.е. запрашивается несколько SCUR блоков. Увеличение количества обрабатываемых блоков, а именно получение на большее количество блоков SCUR блокировки, негативно сказывается на эффективности системы. Однако вероятность каскадного запрашивания блоков UNDO обусловлена только характером трафика. Будем считать, в контексте моделирования GCF, что характеристиками трафика является соотношение запросов SCUR, XCUR и CR*, а значит трафик с большим процентом каскадно запрошенных блоков UNDO, с точки зрения модели, рассматриваемой в этой статье – это трафик, в котором больший процент SCUR запросов по сравнению с XCUR и CR*.

Распишем в таблице 1 алгоритм доступа к блоку, описанный выше, и расставим соответствующие 8 сценариев – без сценария wd, поскольку к времени доступа к блоку в пользовательских сессиях он не имеет отношения:

Табл.1. Выбор сценария в зависимости от состояния и запроса состояния

		Существует в глобальном кэше				Нет
		xcur	scur	cr only	Вылетел	
Запрос состояния	cr*	crp*			-	-
	Xcur	ww	wr	-	rd2	rd1
	Scur	rw	rr	-		

Символом «-» в таблице обозначены запрещённые состояния, т.е. Запрос состояния невозможен при таком текущем состоянии в глобальном кэше. Запрещённые состояния возникают по той причине, что из трафика мы выделили подкласс, обрабатываемый crp алгоритмом. Этот подкласс обладает 2 свойствами: «блок должен быть запрошен в CR состоянии» и «блок запрошен на тот момент времени, на который существует копия блока в локальном кэше». Первое запрещённое состояние получается поскольку CR блок невозможно преобразовать в SCUR или XCUR. Поэтому наличие CR блоков никак не связано с получением SCUR и XCUR блокировок. То есть, если запрашивается XCUR и SCUR проверяется также SCUR и XCUR в глобальном кэше и игнорируется проверка есть ли кроме XCUR и CR состояния блоков. Второе запрещённое состояние получается по причине того, что наличие в локальном кэше (условие cr* состояния) автоматически означает его наличие и в глобальном – т.е. блок не может быть в состояниях «нет» (в кэше) и «вылетел» (из кэша).

Анализ временных задержек в различных сценариях доступа к блоку

Следует указать на то, что факторы локального кэша и локальной мастер-ноды существенно сокращают время обработки. Наличие в локальном (для ноды которая выполняет обработку) в нужном состоянии кэше приводит к тому что в сценариях ww, rr, wr и rw не нужно пересылать сам блок. С другой стороны, если нода, обрабатывающая запрос, является мастер-нодой – то не нужно запрашивать состояние блока по сети.

Распишем таблицу задержек с учетом этих условий. За основу возьмём табличную форму алгоритма, добавив в Таблицу 1 факторы локальной мастер-ноды и локального кэша.

Табл.2 Выбор сценария с учетом фокторов мастер-ноды и локальноко кэша

	Запрос состоя-ния	Существует в глобальном кэше и в каком состоянии					вылетел	Нет
		xcur		scur		cr only		
		локально	нелокал.	локально	нелокал.			
Мастер	cr*	crp*					rd2	rd1
	xcur	ww	ww	wr	wr	-		
		rw	rw	rr	rr	-		
Не мастер	cr*	crp*						
	xcur	ww	ww	wr	wr	-	rd2	rd1
	scur	rw	rw	rr	rr	-		

В таблице выделены сценарии с класс-сическим временем выполнения. Во всех остальных случаях будут отрабатывать сценарии оптимизации, связанные с фактором локальной мастер-ноды и (или) фактором локального кэша. Подставив времена выполнения из формул 1,2,3, 4, 5,6 занесём времена выполнения в таблицу 3, с учётом оптимизаций мастер ноды и локального кэша для оставшегося трафика после выделения специфического cr* подтрафика:

Табл.3. Времена выполнения сценариев

Запрос состояния		Существует в глобальном кэше				Вылетел	нет
		xcur		scur			
		л	г	л	г		
мастер	xcur	0	$t_{net}+t_{send}+t_{fp}$	0	$t_{net}+t_{send}$	$2t_{net}+t_{io}$	t_{io}
	scur	-(0)	$t_{net}+t_{send}+t_{fp}$	0	$t_{net}+t_{send}$		
немаст.	xcur	0	$2t_{net}+t_{send}+t_{fp}$	$2t_{net}$*	$2t_{net}+t_{send}$*	$4t_{net}+t_{io}$	$2t_{net}+t_{io}$
	scur	-(0)	$2t_{net}+t_{send}+t_{fp}$	0	$2t_{net}+t_{send}$		

* Указано время при отсутствии конфликта. При конфликте, сценарий ww и время обработки соответственно: $2t_{net}+t_{send}+t_{fp}$

С точно такой же структурой распишем вероятности попадания в каждое из состояний, на основе вероятностей ключевых факторов:

Табл.4 Вероятности попадания в сценарий

Запрос состояния		Существует в глобальном кэше				вылетел	нет
		xcur		scur			
		Л	г	л	г		
Мастер	xcur	$p_{gc}p_{lc}p_{xcur}$ $p_{rxcur}p_{mn}$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rxcur}p_{mn}$	$p_{gc}p_{lc}p_{scur}$ $p_{rxcur}p_{mn}$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rxcur}p_{mn}$	$(1-p_{gc})p_{out}p_{mn}$	$(1-p_{gc})(1-p_{out})p_{mn}$
	scur	$p_{gc}p_{lc}p_{xcur}$ $p_{rscur}p_{mn}$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rscur}p_{mn}$	$p_{gc}p_{lc}p_{scur}$ $p_{rscur}p_{mn}$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rscur}p_{mn}$		
Не мастер	xcur	$p_{gc}p_{lc}p_{xcur}$ $p_{rxcur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rxcur}(1-p_{mn})$	$p_{gc}p_{lc}p_{scur}$ $p_{rxcur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rxcur}(1-p_{mn})$	$(1-p_{gc})p_{out}$ $(1-p_{mn})$	$(1-p_{gc})(1-p_{out})(1-p_{mn})$
	scur	$p_{gc}p_{lc}p_{xcur}$ $p_{rscur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rscur}(1-p_{mn})$	$p_{gc}p_{lc}p_{scur}$ $p_{rscur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rscur}(1-p_{mn})$		

где p_{gc} – вероятность попадания в глобальный кэш

p_{lc} – вероятность попадания в локальный для ноды обрабатывающей запрос кэш

p_{rxcu} – вероятность того, что запрос будет XCUR.

p_{rscur} – вероятность того, что запрос будет SCUR.

p_{xcur} – вероятность того, что состояние запрошенного блока XCUR

p_{scur} – вероятность того, что состояние запрошенного блока SCUR

p_{mn} – вероятность того, что нода выполняющая запрос является нодой-владельцем (мастер)

p_{out} – вероятность того, что блок оказался в состоянии «вылетел»

Заметим, что поскольку состояния «нет» и «вылетел» определяются p_{gc}, а необходимость использования блока в состоянии «cr only» определяется только во время запросов cr* с вероятностью p_{cr*}, и, поскольку, сценарии рассматривающие срабатывание при вероятностях p_{gc} и p_{cr*} независимы от p_{scur} и p_{xcur} – события нахождения блока в состояниях p_{scur} и p_{xcur} не зависят от p_{gc} и p_{cr*}, а значит и $p_{scur}+p_{xcur}=1$. В контексте того, что после выделения cr* подтрафика из общего трафика, и учитывая что crp запрос распадается на комбинацию rscur и rxcur запросов, имеем $p_{rxcur}+p_{rscur}=1$.

Таким образом, с учетом того, что приведенные выше таблицы 3 и 4 исключают CR* сценарий, имеем 21 сценария, попадание в каждый из которых определяется вероятностями, вычисляемыми по приведенным в таблице 4 формулам на основе 7-ми базовых вероятностей: p_{cr*}, p_{gc}, p_{lc}, p_{rxcur}, p_{xcur}, p_{mn}, p_{out}.

Итоговая формула

Комментируя таблицы 3 и 4 принципиально классифицировать использованные вероятности и времена как базовые

характеристики системы, базовые характеристики трафика и производные характеристики. Базовые характеристики системы это – p_{mn}, t_{net}, t_{send}, t_{io}. Базовые характеристики трафика: p_{rxcur}, p_{rscur} или $(1-p_{rxcur})$, p_{cr*}, p_{pcr} (исключён из формул в модели т.к. может быть представлен комбинацией SCUR и XCUR запросов). И производные характеристики, обусловленные трафиком, системой и алгоритмом доступа к блоку в распределённой системе: p_{xcur}, p_{scur} или $(1-p_{rxcur})$, t_{fp}, p_{gc}, p_{out}, p_{lc}, p_{down}.

Заметим, что хотя интенсивность и не фигурирует в формулах времени доступа напрямую, чаще всего она является основой для вычисления производных параметров. В более широком смысле характеристика трафика – это вектор XCUR, SCUR и CR интенсивностей запросов к каждому ресурсу системы и правила определяющие динамику этих интенсивностей включая правила обратной связи.

Также отметим, что поскольку в формулах не отражен сценарий запись на диск. Задержки отложенной записи, т. е. работа процесса DBWn не включаются в задержки пользовательских сессий, однако могут конкурировать за ресурсы, создавая задержки. При стационар-ной работе системы это допустимо, но при анализе устойчивости на коллапс, этим факто-ром пренебрегать нельзя.

Перемножив из таблиц 3 и 4 соответствующие вероятности на соответствующие времена получаем итоговую формулу для вычисления среднего времени доступа:

$$
\begin{aligned}
t_{acc} = (1-p_{cr*})(\ & p_{gc}(1-p_{lc})p_{xcur}p_{rxcur}p_{mn}\left(t_{net}+t_{send}+t_{fp}\right)+\\
&+p_{gc}(1-p_{lc})p_{xcur}(1-p_{rxcur})p_{mn}\left(t_{net}+t_{send}+t_{fp}\right)+\\
&+p_{gc}(1-p_{lc})p_{xcur}p_{rxcur}(1-p_{mn})\left(2t_{net}+t_{send}+t_{fp}\right)+\\
&+p_{gc}(1-p_{lc})p_{xcur}(1-p_{rxcur})(1-p_{mn})\left(2t_{net}+t_{send}+t_{fp}\right)+\\
&+p_{gc}p_{lc}p_{scur}p_{rxcur}p_{mn}2t_{net}+p_{gc}(1-p_{lc})p_{scur}p_{rxcur}p_{mn}(t_{net}+t_{send})+\\
&+p_{gc}(1-p_{lc})p_{scur}(1-p_{rxcur})p_{mn}(t_{net}+t_{send})+\\
&+p_{gc}(1-p_{lc})p_{scur}p_{rxcur}(1-p_{mn})(2t_{net}+t_{send})+\\
&+p_{gc}(1-p_{lc})p_{scur}(1-p_{rxcur})(1-p_{mn})(2t_{net}+t_{send})\ )+p_{cr*}\cdot 0
\end{aligned}
$$

Это время и является при неизменных базовых характеристиках системы и трафика критерием, насколько техники оптимизации, внесённые в базовый алгоритм, эффективны. Идеальной является ситуация, когда $t_{acc} = 0$. И снижение t_{acc} является целевой функцией систем с распределённой памятью использующих принципы GCF.

Заключение

Предложенная модель позволяет при различных трафиках вычислять среднее время доступа к блоку в распределённых кластерных системах использующих как основу алгоритм GCF. Необходимость построения модели обусловлена требованиями возможности оценки эффективности различных техник оптимизации алгоритма GCF.

Однако, стоит указать также на направления требующие дополнительного исследования. В первую очередь, это методика описания базовых характеристик трафика и методика вычисления производных параметров.

Среди недостатков модели стоит указать, что не учитываются задержки принятия решения о выборе сценария. Предполагается, что время обмена информацией между нодами существенно больше времени принятия решения о выборе сценария на каждом шаге алгоритма. Это условие справедливо, кроме случаев, когда скорость обмена информацией между нодами сопоставима с временем обработки. В современных системах это справедливо, но с учётом развития технологий коммуникации модель может потерять свою актуальность.

Также следует обратить внимание на то, что процесс каскадного запрашивания блоков UNDO – представляет интерес в контексте анализа системы на коллапс по блокировкам, поскольку обработка блока не завершается до тех пор, пока все вспомогательные блоки UNDO не будут прочитаны. Причем какие блоки UNDO и сколько их, мы можем узнать

только прочитав первый блок с SCUR запросом. То, что необходимость получения очередного блока может быть выявлена только после прочтения предыдущего, приводит к увеличению средней последовательной цепочки блоков при обработке запроса, т. е. обработка блока начинается только по завершению предыдущего. Характер траффика, а именно наличие часто изменяемых т.н. «горячих» блоков определяет среднюю длину цепочки. В предложенной модели, поскольку обработка CR блока распадается на обработку нескольких SCUR блоков, рассматривался весь трафик как набор обработки SCUR, XCUR и CR* блоков. Учитывая, что в OLTP системе запрос выполняется процессом без распараллеливания, рассматривалось среднее количество блоков обрабатываемых запросом, и пренебрегалось зависимостью обработки блоков в цепочке. Оптимизация этого последовательного подхода представляет интерес, однако выходит за рамки данной статьи и требует дополнительного исследования.

Кроме того, предложенная модель может быть использована не только в shared-everything архитектуре, но и в топологиях, в которых доступ к памяти организован как в GCF, а диски на каждой системе собственные, т. е. для моделирования распределенных систем, использующих общую память и работающих с общими ресурсами в глобальных сетях.

Список литературы

1. Гусев Е.И. Исследование области применения неблокирующего алгоритма фиксации распределённых транзакций // Вісник НТУУ "КПІ". Сер. Інформатика, управління та обчислювальна техніка. - 2012. Випуск 57. - С.76 – 80
2. Гусев Е.И. Исключение блокирования общего ресурса в распределённых системах / Гусев Е.И.; Кулаков А.Ю. // Вісник НТУУ "КПІ".Сер. Інформатика, управління та обчислювальна техніка. - 2011. Випуск 54. - С.162 – 166.
3. Oracle® Real Application Clusters Administration and Deployment Guide 11g Release 2(11.2) E16795-08
4. Lampson B., Sturgis H. "Crash recovery in a distributed data storage system," Tech. Rep., Computer Science Lab., Xerox Palo Alto Research Center, Palo Alto, Calif., 1976
5. Menasce D. A., Popek G. J., Muntz R. R. "A locking protocol for resource coordination in distributed databases," ACM Trans. Database Syst. 5, 2 (June 1980), 103-138.
6. Riyaj Shamsudeen. RAC Hack: Deep review of LMS/LGWR process. (July 2011). [Электронный ресурс] . – Режим доступа: http://orainternals.wordpress.com/2011/07/