УДК 004.318

Применение кэша и справочника DMA-обменов в NUMA-системах для повышения производительности подсистемы ввода-вывода

Исаев М.В., Поляков Н.Ю.

Рассматривается проблема упорядоченного исполнения DMA-операций. Описывается механизм использования кэш-памяти для DMA обменов, как один из способов решения данной проблемы. Предлагается метод оптимизации служебного трафика NUMA-системы, содержащей кэш-памяти для DMA-обменов.

Ключевые слова: NUMA, DMA, DMA-кэш, директория, когерентность, консистентность.

Одним из основных способов повышения производительности современных вычислительных комплексов является увеличение количества ядер в многопроцессорной системе. Стандартом де-факто для таких систем стало объединение процессоров на общей памяти с разнородным временем доступа (NUMA-архитектура, Non-Uniform Memory Architecture). Однако их производительность ограничена пропускной способностью подсистемы ввода/вывода, которая определяет скорость, с которой процессор может принимать внешние данные для обработки. В современных процессорах скоростной обмен данными с внешними устройствами осуществляется, в основном, посредством прямого доступа устройств в оперативную память – DMA (Direct Memory Access). В режиме DMA могут передаваться данные от DSP-сопроцессора, от внешней антенны после обработки в АЦП, от сетевой карты, жесткого диска и других устройств.

В статье будут рассмотрены способы решения двух задач. Первая задача – упорядоченное исполнение DMA-операций записи в память NUMA системы, необходимое для корректной работы драйверов периферийных устройств. Вторая задача – совместно с решением первой задачи оптимизировать служебный трафик системы.

Проблема упорядоченности DMA-обращений

Особенностью некоторых периферийных интерфейсов (например, PCI Express) является то, что операции DMA- записи данных в память ЦП имеют тип posted (почтовый), то есть считаются завершенными сразу после выдачи из устройства, а DMA-обмен, состоящий из posted операций, - после выдачи последней операции записи в обмене. По окончании обмена устройство либо меняет значение своего статусного регистра, либо значение некоторой ячейки (дескриптора обмена) в памяти СнК (посредством DMA-записи).

Перед использованием данных, записанных в память ЦП, программист/программа проверяет, завершился ли обмен, считывая либо статусный регистр устройства, либо дескриптор обмена. Отсутствие контроля завершения операций DMA-записи в NUMA-системе может привести к считыванию из памяти некорректных данных и, как следствие, к некорректной работе программы. В зависимости от способа индикации завершенности обмена (статусный регистр или дескриптор) можно выделить 2 сценария возникновения ошибки. В первом случае DMA-обмен завершается в памяти ЦП позднее, чем ЦП прочтет статусный регистр и начнет читать записанный в память массив, то есть, происходит переупорядочивание операций DMA-записи элемента массива и процессорного чтения регистра устройства. Во втором случае меняется порядок выполнения операции DMA-записи элемента массива и DMA-записи дескриптора.

Оба описанных события могут иметь место в NUMA-системе ввиду того, что память физически распределена между несколькими процессорами. Так время доставки запроса по DMA-записи до памяти, физически подключенной к процессору A (далее – память процессора A), может оказаться больше суммарного времени доставки DMA-запроса по записи в память процессора B и запроса и ответа на процессорное чтение из памяти процессора A, что соответствует второму случаю.

В соответствии с описанными случаями возникают две задачи. Первая задача – обеспечение упорядоченности DMA-записей относительно ответов на процессорные чтения статусных регистров периферийных устройств. Вторая задача - обеспечение упорядоченности операций DMA-записи друг относительно друга (консистентность памяти).

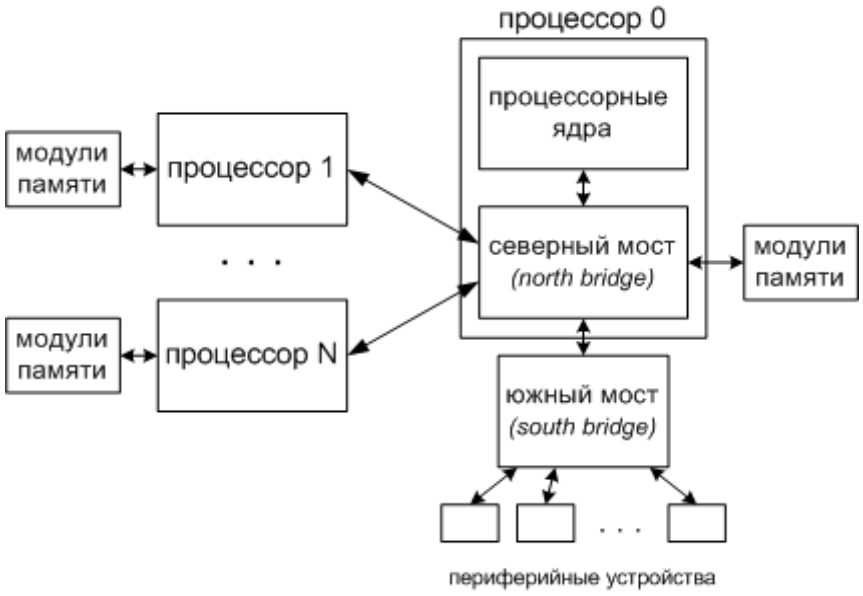

Рис. 1. NUMA-система

DMA-кэш

В NUMA-системах гарантировать упорядоченность двух DMA-записей в памяти разных процессоров можно только при условии наличия средств контроля завершения этих операций. Таким образом, в некоторой точке системы (назовем её точка упорядоченности) DMA-запросы на запись должны становиться непочтовыми (non-posted). Через точку упорядоченности должны проходить также все ответы на процессорные чтения. Такой точкой может быть либо Южный мост, либо Северный мост процессора, к которому подключено периферийное устройство (***Рис. 1***).

Наиболее очевидным решением вышеописанных задач является выполнение DMA-записей ***атомарно***, т.е. очередная DMA-запись или ответ на процессорное чтение отправляется только после полного завершения предыдущей DMA-записи. Данный способ надежен, но неприменим к системам, требующим высоких скоростей обмена, т.к. отсутствует конвейерность выполнения операций.

Принцип консистентности памяти в применении к процессорным операциям, реализуемым в соответствии с политикой write back кэш-памяти, достигается следующим образом. При выполнении ядром когерентных записей в общую память системы сначала проверяется наличие и состояние данных в своей кэш-памяти и кэш-памяти каждого из других ядер. Запись допускается только после того, как целевая строка памяти окажется в кэш-памяти ядра, а ядро получит права на модификацию строки [1]. В данном случае точкой упорядоченности служит кэш-память ядра, так как несмотря на то, что после выполнения записи кэш-строка может и не выталкиваться в память, все последующие чтения по адресу этой строки будут возвращать данные записи в силу когерентности системы.

Подобную схему можно использовать для выполнения DMA-записи. Сначала из точки упорядоченности отправляется запрос на владение целевой строкой, который ведет к ее вычеркиванию из кэш-памяти всех ядер. Далее в точке упорядоченности данные записи, пришедшие из периферийного устройства, накладываются по маске на подложку (ответ на запрос владения), и модифицированная строка отправляется в память. Получается своего рода кэш-память (далее – DMA-кэш), которая участвует в механизме поддержания когерентности подсистемы памяти. DMA-запись будет завершена в момент модификации подложки, ибо в ответ на процессорные чтения по данному адресу DMA-кэш вернет модифицированные данные.

Для достижения упорядоченности DMA-операций наложение данных на подложку должно выполняться в порядке прихода DMA-запросов. Так если для некоторого запроса данные ещё не модифицированы (вследствие незавершенности предыдущих запросов или задержки в получении владения строкой), то в ответ на процессорные чтения по адресам,

совпадающим с адресами последующих DMA-запросов, будет возвращаться либо сообщение об отсутствии данных (если владение не получено), либо подложка (если владение получено). Ответы на процессорные чтения также должны выдаваться из точки упорядоченности после модификации данных всех предыдущих DMA-запросов.

Существенным преимуществом данного метода является возможность конвейерного исполнения запросов: 1) нет необходимости перед отправлением запроса за владением дожидаться завершения предыдущих запросов и 2) запросы по записи модифицированной строки можно отправлять в любом порядке, т.к. результат записи уже виден системе.

К недостаткам метода относится увеличение трафика каналов межпроцессорных связей при запросах в память других процессоров за счет передачи подложки в точку упорядоченности DMA-запросов. Данная проблема решается для обменов целыми кэш-строками (большими массивами данных) отправкой запроса за владением, не требующего подложки. Другим недостатком метода является увеличение количества кэшей в системе, что в системах, поддерживающих когерентность памяти путем полного снупирования, приводит к значительному увеличению служебного трафика. Решение данной проблемы предлагается в следующей главе.

Оптимизация когерентного DMA-траффика в NUMA-системах

В том случае, если DMA-записи выполняются описанным выше способом, DMA-кэш в первом приближении можно считать независимым абонентом подсистемы памяти. Таким образом, логично было бы решать проблему когерентного доступа по DMA к общей памяти теми же средствами, что и для процессорных ядер. Однако у DMA-кэшей есть свои особенности, которые позволяют найти для них более эффективные методы.

В многопроцессорных NUMA-системах межпроцессорная когерентность может поддерживаться одним из двух способов: либо с помощью широковещательного опроса всех абонентов системы, либо с помощью справочника (директории), содержащего информацию о возможности нахождения строки памяти в кэшах системы и снижающего когерентный трафик в системе, рассылая запросы только некоторым процессорам. Системы, основанные на широковещательном протоколе являются более простыми в разработке и более надёжными, однако при увеличении количества процессоров, количество служебного когерентного трафика в системе растёт квадратично, что является сдерживающим фактором при масштабировании таких систем. Появление DMA-кэша в каждом процессоре системы фактически удваивает количество абонентов, что существенным образом сказывается на скорости обработки всех запросов системы в общую память.

Для снижения когерентного трафика обычно используются директории двух видов: либо полные по памяти, либо частичные - располагающиеся в отдельной кэш-памяти директории [2]. В директории размещается информация о том, какими строками владеет каждый процессор в случае частичной директории, и о том, какие процессоры владеют каждой строкой памяти в случае полной. Однако для DMA-запросов использование таких структур является неэффективным с точки зрения объёмов памяти, затрачиваемой на хранение служебной информации.

Особенность DMA-кэша состоит в том, что строка заводится в нем для выполнения только одной записи и, в отличие от строки в процессорном кэше, не переиспользуется и вытесняется сразу после модификации. В связи с этим объем DMA-кэша выбирается в зависимости от максимального темпа обмена и среднего времени выполнения одной операции и должен иметь объем на несколько порядков меньший объема процессорных кэш-памятей (2 КБ для скорости 8 ГБ/с и времени выполнения операции 256 нс).

Если при этом использовать полную директорию для хранения информации о наличии строк памяти в DMA-кешах, то будут задействованы объёмы памяти, значительно превосходящие объёмы DMA-кэшей (1 Б на каждые 128 Б данных, или 1 ГБ для системы с 4 процессорными узлами по 32 ГБ памяти в каждом). В частичной директории информация о нахождении строки в DMA-кэшах будет занимать гораздо меньше места, однако, в силу особенности организации частичных директорий, добавление информации о DMA-кэшах может увеличить требуемый для директории объём памяти на 30%. Так, если в строке директории размером 32 бита информация о строках памяти, находящихся в удалённых процессорах, занимает 12 бит, то добавление информации о DMA-кэшах займёт ещё 10 бит или 30% [3].

В силу особенностей организации DMA-кэшей, наиболее эффективной структурой для хранения информации о размещении кэш-строк в DMA-кэшах является полная по кэшу директория. Данный тип директорий редко используется для процессорных кэшей ввиду их значительного объёма, однако данный способ оказывается самым эффективным для DMA-кэшей в силу их небольшого объема последних. Так для 4 DMA-кэшей по 2 КБ объем директории составит несколько сотен байт.

Директория DMA-кэша возвращает служебный трафик, связанный с когерентным доступом ядер в память на уровень системы без DMA-кэшей. Среди недостатков такого подхода следует отметить необходимость полноты директории. Так как каждый DMA-кэш может быть полностью занят работой с памятью одного узла, то директории в каждом узле должны быть рассчитаны на полный объём всех DMA-кэшей в системе. Таким образом, с ростом количества DMA-кэшей в системе, объём памяти DMA-директорий растёт

квадратично. Однако при значительном увеличении объёма и количества DMA-кэшей всё ещё остаётся возможным использовать информацию о наличии строк памяти в DMA-кэшах в частичной процессорной директории, заведя туда информацию о DMA-кэшах.

Заключение

Использование в NUMA-системах DMA-кэша совместно со справочником для него позволяет выполнять операции DMA-записи в первоначальном порядке, сохраняя при этом высокий темп обмена за счет конвейерной обработки запросов и прежний уровень служебного трафика (по сравнению с системами без DMA-кэша), связанного с когерентным доступом процессорных ядер в память. Стоит отметить, что данный метод позволяет решать проблему упорядоченности независимо от ПО, типа периферийного устройства и конфигурации системы.

Литература

1. Intel 64 and IA-32 Architectures Software Developer’s Manual, Volume 3A: System Programming Guide, Part 1. – 2012.
2. *Исаев М.В.* Анализ способов эффективной оптимизации внешнего протокола когерентности на базе когерентной директории. - Труды 54-й научной конференции МФТИ, Радиотехника и кибернетика, том 1, с. 18-19, Москва-Долгопрудный-Жуковский, 2011.
3. *Вараксин В.Н., Исаев М.В., Сахин Ю.Х.* Оптимизация межпроцессорного протокола когерентности с помощью справочника в микропроцессоре Эльбрус-4С+. - “Вопросы радиоэлектроники”, сер. ЭВТ, 2013, вып. 3, с. 14-26.