УДК 004.318

Обеспечение упорядоченности выполнения DMA-операций записи в NUMA системах методом отмены

Н.Ю. Поляков

ЗАО «МЦСТ»

ОАО «Институт электронных управляющих машин имени И.С. Брука»

В настоящее время для обмена данными между центральным процессором и периферийными (внешними) устройствами используется механизм прямого доступа в память процессора - Direct Memory Access (DMA). Это относится к таким устройствам как жесткий диск, сетевая карта, различные PCI устройства и др. Особенностью некоторых периферийных интерфейсов (например, PCI Express) является то, что операции DMA-записи считаются завершенными сразу после их выдачи из устройства. В NUMA-системах данная особенность может привести к некорректному завершению DMA-обмена вследствие переупорядочивания DMA-операций записи [1,2].

Существует несколько методов решения данной проблемы, наиболее распространенным из которых является введение барьерных операций [3]. Однако он неприменим, если драйвер внешнего устройства не использует барьерные операции и отсутствует его исходный код. Такого недостатка лишен механизм кэширования [2,4], но наличие дополнительной кэш-памяти (DMA-кэша) приводит, во-первых, к возрастанию служебного трафика, необходимого для поддержания когерентного состояния памяти, и, во-вторых, к возрастанию времени выполнения DMA-операции записи, так как каждая операция распадается на две последовательные (запрос на владение и сама запись).

Предлагается новый способ обеспечения упорядоченного выполнения DMA-операций записи – *метод отмены*. Он применим к тем NUMA-системам, где проверка когерентности для DMA-операций записи выполняется в процессоре, к которому подключена целевая память. Кроме того, требуется, чтобы выполнение DMA-операции записи в *точке упорядочивания* [2] было разделено на следующие этапы: 1) отправка информационной части запроса в точку обработки (сериализации) запросов в память (например, системный коммутатор), 2) ожидание запроса за данными записи и сообщения о завершении действий по проверке когерентности и 3) выдача данных записи.

Этот метод предполагает после выполнения этапа 1 помещать поступающие DMA-запросы в некоторую очередь, расположенную в точке упорядочивания. Далее, на этапе 2, все находящиеся в очереди запросы ожидают перехода в *состояния упорядоченности*. Запрос считается *упорядоченным*, если для него и всех предшествующих запросов

получено сообщение о завершении действий по проверке когерентности. Этап 3 выполняется только для упорядоченных запросов, неупорядоченные запросы, для которых получен запрос за данными, ожидают перехода в состояние упорядоченности. Запросы выдаются из очереди только после перехода в состояние упорядоченности.

Без дополнительных доработок описанный выше алгоритм может привести к зависанию NUMA-системы, описанному в [2]. Метод отмены позволяет избежать этого зависания. Предлагается для каждой DMA-операции, получившей запрос за данными, запускать счетчик времени ожидания перехода в состояние упорядоченности. Если в течение заранее определенного времени упорядочивания не произошло, данные выдаются с признаком отмены записи или нулевой маской записи и этап 1 для данного запроса повторяется.

Преимуществами предлагаемого метода является, во-первых, независимость от программного обеспечения и, во-вторых, отсутствие необходимости добавления кэш-памяти, что актуально для NUMA-систем, использующих для поддержки когерентного состояния памяти широковещательный опрос кэш-памятей.

Литература

1. *Поляков Н.Ю.* Проблема упорядоченности DMA-операций в NUMA-системах и способы её решения // 55-я научная конференция МФТИ. – 2012.

2. *Перов Д.Ю., Поляков Н.Ю.* Обеспечение упорядоченности выполнения DMA-операций в NUMA-системах методом предварительного кэширования // Вопросы радиоэлектроники. Серия ЭВТ. – 2013. – Вып. 3. – С. 38-47.

3. *Shuwei Bai [et al.]* Barrier Synchronization for CELL Multi-Processor Architecture // First IEEE International Conference on Ubi-media Computing. – 2008. – P. 155-158.

4. *Debendra Das Sharma* Intel 5520 Chipset: An I/O Hub Chipset for Server, Workstation, and High End Desktop // Hot Chips 21. – 2009. – Session 2.