МОДЕЛЬ ОРГАНИЗАЦИИ ОБМЕНА ДАННЫМИ

В МНОГОПРОЦЕССОРНОЙ ТЕЛЕКОММУНИКАЦИОННОЙ СИСТЕМЕ

А.Койнов
akoynov@infinetwireless.com

Многопроцессорные решения все чаще используют в современных телекоммуникационных системах. Это обусловлено главным образом двумя факторами [1]: возможностью практически неограниченного увеличения производительности системы и особенностями решаемых системой задач, когда в одном устройстве требуется использовать несколько различных процессоров. Однако для достижения высокой производительности таких систем нужно обеспечить эффективный межпроцессорный обмен данными. Существующие программные средства организации обмена не решают эту задачу в полной мере. В настоящей статье рассматривается модель межпроцессорного обмена, которая позволяет значительно повысить производительность многопроцессорных телекоммуникационных систем.

ПРЕДПОСЫЛКИ РАЗРАБОТКИ МОДЕЛИ

Организация межпроцессорного обмена в многопроцессорных телекоммуникационных системах, как правило, осложняется следующими факторами:

- через межпроцессорный канал нужно передавать множество мелких пакетов с данными;
- время на передачу каждого пакета строго ограничено;
- передача ведется через относительно медленный межпроцессорный канал;
- выбор вычислительных средств ограничен экономическими факторами, условиями эксплуатации оборудования, требованиями по энергопотреблению и т.п.

Классическая модель обмена состоит из следующих основных этапов [2].

1. Задача, принимающая данные, вызывает функцию получения данных драйвера межпроцессорного обмена, указывая адрес буфера, в который будут скопированы данные, его размер и идентификатор транзакции; после вызова функции выполнение задачи блокируется до окончания обмена или истечения таймаута транзакции.

2. Задача, передающая данные, вызывает функцию посылки данных, указывая адрес буфера, из которого берутся данные, его размер и идентификатор транзакции; после вызова функции задача блокируется до завершения операции или истечения таймаута.

3. Драйвер межпроцессорного обмена на передающей стороне, как правило, посредством прерывания, запрашивает у принимающей стороны буфер с переданным ему идентификатором.

4. Драйвер принимающей стороны, получив запрос от передающей стороны, при наличии буфера для транзакции с заданным идентификатором выдает его адрес, извещая удаленную сторону посредством прерывания.

5. После получения адреса приемного буфера передающая сторона запускает копирование данных посредством DMA* или под управлением процессора (программно-управляемый обмен).

6. По окончании обмена передающая сторона извещает принимающую посредством выдачи прерывания; передающей и принимающей задачам возвращается управление.

К достоинствам классической модели можно отнести ее простоту и известность в кругу разработчиков.

К недостаткам следует причислить существенные ограничения производительности системы, привносимые моделью. На уровне операционной системы ограничения связаны с блокировкой задач на время межпроцессорных транзакций [3]. На аппаратном уровне производительность снижается за счет большого количества прерываний, выдаваемых и обрабатываемых при проведении каждой транзакции [4] (до четырех прерываний при использовании DMA). Эффект от использования DMA в данном случае не гарантирован. Во-первых, в случае

*DMA (Direct Memory Access) – подсистема прямого доступа к памяти. В большинстве современных вычислительных архитектур данная подсистема позволяет осуществлять ввод и вывод данных параллельно работе процессорного ядра, не задействуя его.

транзакций малой длины (до сотен байт) время на подготовку транзакции и обработку прерываний по ее окончании сопоставимо с временем, которое требуется на проведение транзакции в режиме программно-управляемого обмена. Во-вторых, модель не позволяет в явном виде использовать цепочечные режимы работы DMA, в которых без вмешательства ядра может быть произведено несколько транзакций подряд.

Известен ряд способов преодоления ограничений, привносимых классической моделью.

Для устранения простоя задач обмен может вестись в неблокирующем режиме с ограничением доступа к буферам до окончания межпроцессорной транзакции. Извещение об окончании производится посредством функций-нотификаторов (функций, извещающих приложения об определенных событиях при обмене) или путем опроса драйвера межпроцессорного обмена (что нежелательно, ибо чревато дополнительными накладными расходами на операции опроса).

Наименьшие накладные расходы на работу DMA достигаются при использовании цепочечных режимов контроллера. Задействовать их оптимально возможно, если межпроцессорный обмен происходит асинхронно. При этом транзакция выполняется в два действия: сначала буфер помещается в очередь, а затем инициируется обмен. В результате за одну транзакцию будут переданы все буферы, находящиеся в очереди.

Кроме того, имеет смысл выделить различные программные механизмы (которые в конечном итоге могут опираться на различные аппаратные механизмы передачи данных через физический межпроцессорный канал) для реализации управляющих транзакций и передачи основного потока данных. Такое разделение обусловлено различиями в количествах передаваемых данных, приоритете, допустимых задержках при передаче.

Дальнейшая оптимизация обычно ориентирована на работу с буферной памятью: очень важно минимизировать частоту запросов на динамическое выделение памяти. Стоит заметить, что во многих случаях межпроцессорные транзакции носят периодический характер, и параметры потока данных известны заранее. Оптимизировать периодические транзакции можно, введя в модель межпроцессорного обмена понятие «логического межпроцессорного канала», в рамках которого буферная память будет распределяться статически при его создании.

Также следует обратить внимание на распространенный случай транзитной пересылки данных, когда, например, центральный процессор получает данные из сети и пересылает их в сопроцессор. В этом случае целесообразно отправлять данные в память сопроцессора в том же самом контейнере, в котором они были получены из сети, чтобы избежать лишнего копирования данных. Это возможно при условии, что библиотека межпроцессорного обмена позволяет работать с буферной памятью, выделяемой вне библиотеки.

Существующие программные средства организации межпроцессорного обмена можно разделить на три категории:

- многопроцессорные операционные системы (RTEMS, 3L Diamond);
- библиотеки межпроцессорного обмена, созданные на основе универсальных спецификаций (MPI, PVM);
- специализированные библиотеки межпроцессорного обмена (PolyCore Software Poly-Messenger, TI DSP/BIOS Link, CWC IPC Library).

Сравнение характеристик вышеозначенных программных средств (см. таблицу) показывает, что все доступные средства организации обмена в многопроцессорных системах не содержат большей части оптимизаций, необходимых для эффективной работы многопроцессорных телекоммуникационных систем. Вот почему нужна новая модель организации обмена данными. К этой модели предъявляются следующие требования:

- она должна содержать все оптимизации классической модели, перечисленные выше;
- в общем случае модель должна быть предназначена для несимметричной и неоднородной вычислительной системы (в составе системы могут быть различные процессоры с различными операционными системами);

Характеристики ПО для организации межпроцессорного обмена

Программное средство	Оптимизации классической модели межпроцессорного обмена					
	Механизмы для передачи управляющего потока и потока данных	Неблокирующий обмен	Асинхронный обмен	Назначение нотификаторов	Использование внешних буферов	Возможность оптимизации периодических транзакций
RTEMS	-	-	-	-	-	-
3L Diamond	-	-	-	-	-	-
MPI	-	+	+	-	+	-
PVM	-	-	-	-	-	-
Poly-Messenger	-	-	-	-	-	-
CWC IPC Library	+/-	+/-	-	+	+	-
DSP/BIOS Link	+	+/-	-	-	-	+

+ – наличие; - – отсутствие; +/- – частичная реализация.

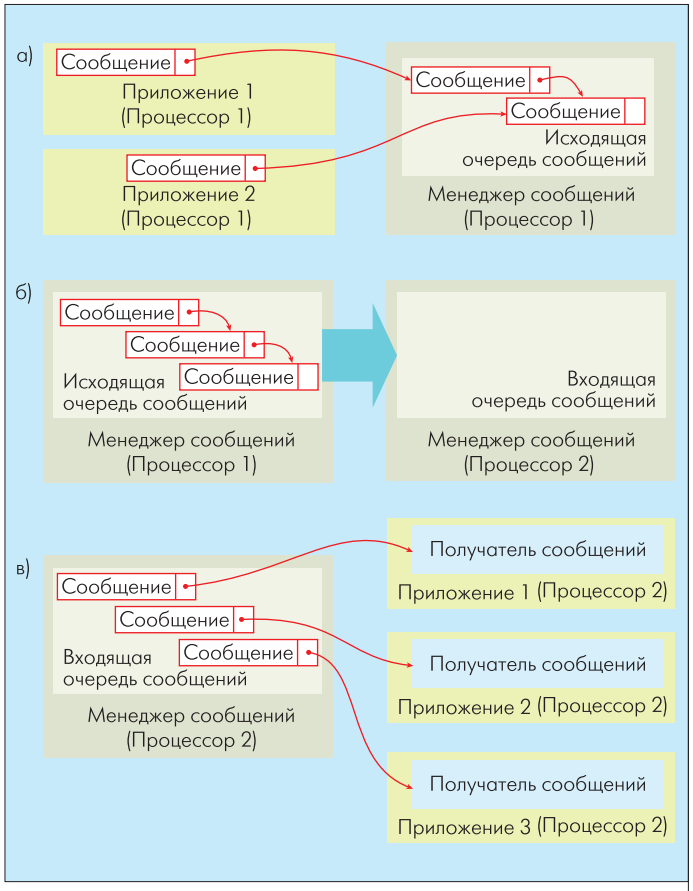

Рис.1. Передача сообщений: а – помещение сообщения в очередь, б – передача очереди сообщений менеджеру сообщений на другом процессоре, в – выдача сообщений получателям

- модель должна быть абстрагирована от конкретных механизмов передачи данных через физический межпроцессорный канал, но вместе с тем должна иметь возможность эффективной реализации на современных аппаратных платформах;
- модель должна быть масштабируема.

ОПИСАНИЕ МОДЕЛИ

На основании указанных требований, имеющегося опыта и анализа существующих библиотек и спецификаций автором статьи была разработана модель межпроцессорного обмена, в основе которой лежат два базовых механизма:

- обмен сообщениями;
- обмен контейнерами через межпроцессорные каналы обмена данными.

Сообщение (Message) представляет собой структуру фиксированного размера, содержащую поле с указанием получателя и типа сообщения, а также несколько полей с данными. Сообщения предназначаются для синхронизации и управления работой приложений, действующих на разных процессорах. Управление обменом сообщениями происходит посредством *менеджера сообщений (Message Manager).*

В процессе обмена менеджер сообщений оперирует не отдельными сообщениями, а их очередью. Сообщение передается приложению, работающему на другом процессоре, асинхронно, без блокировки работы приложений. Процесс имеет две фазы: (1) сообщение помещается в очередь сообщений, ожидающих отправки (рис.1а); (2) очередь сообщений передается менеджеру сообщений на другом процессоре для доставки их получателям (рис.1б).

Таким образом, сообщение передается другому процессору либо немедленно после его помещения в очередь отправляемых сообщений (если нужно доставить его максимально быстро), либо передача сообщений привязана к определенным событиям в системе (например, сообщения могут отправляться по таймеру).

Приложение принимает, интерпретирует и обрабатывает сообщения посредством *получателя сообщений (Message Recipient,* рис.1в). Каждый из получателей имеет уникальный идентификатор, максимальное количество получателей определяется на этапе компиляции.

Контейнер (Storage) представляет собой логическую структуру, состоящую из заголовка и буфера, в котором передаются данные. Контейнер служит средством обмена данными между приложениями, работающими на разных процессорах. В процессе работы контейнер может находиться в трех состояниях:

- заполненный – в буфере, связанном с контейнером, находятся данные, предназначенные приложению-получателю;
- пустой – в буфере контейнера нет данных для получателя;
- свободный – с контейнером не связан буфер.

Приложение ведет обмен контейнерами через логический межпроцессорный *канал обмена данными (Channel).* В одном физическом межпроцессорном канале может работать множество логических каналов. Каналы являются двунаправленными, каждый из них имеет уникальный идентификатор. Приложения могут назначать каналам обработчики для основных событий, связанных с процессом их работы: отправка или получение данных, запуск или остановка канала и т.п. При создании канала приложение указывает количество контейнеров, что позволяет статически распределить память, связанную с каналом и контейнерами. Канал имеет две очереди, в которых хранятся принадлежащие данному каналу контейнеры, ожидающие обработки приложениями: входящие очереди пустых и заполненных контейнеров. Кроме этого, имеется очередь свободных контейнеров, в которой хранятся контейнеры канала, не связанные с буфером.

Управлением процессом обмена данных занимается *менеджер каналов (Channel Manager).* Менеджер содержит четыре очереди для контейнеров:

- исходящие очереди пустых и заполненных контейнеров. Заполняются контейнерами, поступающими из каналов;
- входящие очереди пустых и заполненных контейнеров. Наполняются менеджерами каналов контейнерами, соответственно, из которых и в которые были скопированы данные. Из этих очередей контейнеры попадают в соответс-

Рис.2. Обмен контейнерами: а – заполнение контейнера, б – помещение контейнеров в очереди, в – обмен данными, г – помещение обработанных контейнеров в очереди каналов

твующие входящие очереди каналов, которым они принадлежат.

При передаче или приеме порции данных приложение заполняет заголовок контейнера (рис.2а), указывая местоположение и длину буфера, и передает каналу заполненный или пустой контейнер соответственно. По умолчанию, буферы контейнерам назначает приложение. Однако можно модифицировать канал таким образом, чтобы буферная память выделялась самим каналом при его создании. Далее контейнер подлежит отправке, которая осуществляется в три фазы и, как и в случае передачи сообщений, не блокирует работу приложений.

1. Контейнер помещается в одну из исходящих очередей менеджера каналов (рис. 2б): в очередь контейнеров с пустыми буферами (для приема данных) или в очередь контейнеров с заполненными буферами (для передачи данных).

2. Один из менеджеров инициирует передачу данных через физический канал. В процессе передачи данные из заполненных контейнеров копируются в пустые контейнеры того же канала, принадлежащие удаленной стороне (рис.2в). После завершения обмена данными пустые контейнеры, ставшие заполненными, попадают во входящую очередь заполненных контейнеров менеджера каналов, а контейнеры, ставшие пустыми, попадают во входящую очередь пустых контейнеров.

3. Контейнеры передаются менеджерами каналов из своих входящих очередей во входящие очереди каналов, которым контейнеры принадлежат (рис.2г).

Далее в каждом из каналов, во входящие очереди которых поступили контейнеры, вызываются функции-нотификаторы, сообщающие приложениям о поступлении контейнеров.

Библиотека, созданная на основе предлагаемой модели, с точки зрения архитектуры программного обеспечения состоит из трех частей (рис.3): абстрактной, низкоуровневой аппаратно-зависимой и высокоуровневой прикладной. Абстрактная часть реализует базовую логику работы всех основных компонентов системы. Низкоуровневая часть библиотеки отвечает за взаимодействие с операционной системой, межпроцессорным каналом, периферией, обеспечивающей передачу данных и т.п. Высокоуровневая часть библиотеки предназначена для адаптации к нуждам использующих ее приложений. Это происходит посредством модификации классов, взаимодействующих с приложениями, таких как каналы, получатели сообщений и т.п. Приложения могут назначать обработчики основным событиям, связанным с меж-

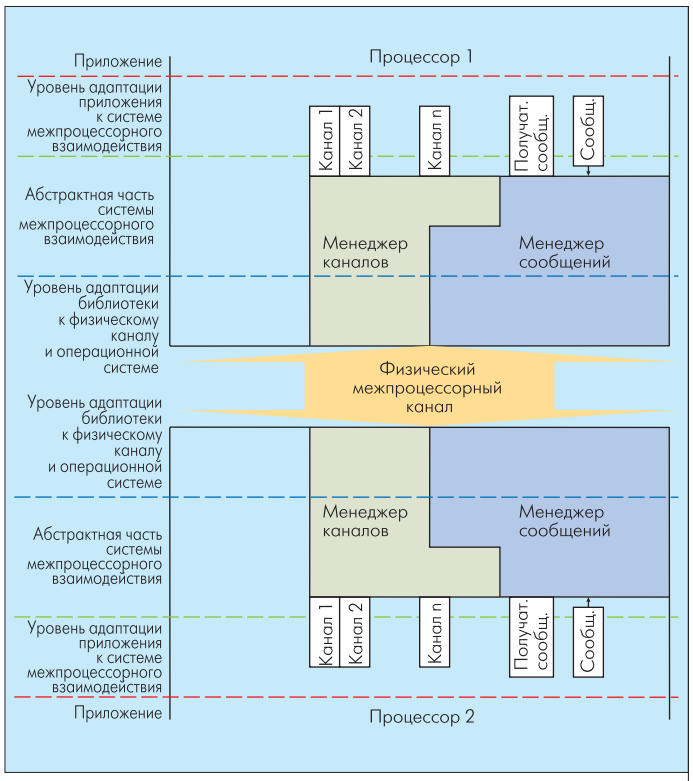

Рис.3. Архитектура системы межпроцессорного взаимодействия

процессорным обменом, управлять распределением буферной памяти и т.п.

ПРИМЕНЕНИЕ МОДЕЛИ

На основе описанной модели была разработана библиотека межпроцессорного обмена IPC Tools. Библиотека используется в оборудовании IP-телефонии и беспроводного широкополосного доступа, которое производит компания InfiNet Wireless.

В указанных платформах в качестве центральных процессоров применяют процессоры IBM PowerPC 405GPr (400 МГц), Intel XScale IXP425 и IXP465 (533 МГц), Intel Xeon (2,4 ГГц) под управлением операционной системы WANFleX. В качестве сопроцессоров используют DSP-процессоры TI TMS320C6415 (600–1000 МГц) под управлением ядра реального времени DSP/BIOS. Во всех случаях физическим межпроцессорным каналом является шина PCI.

Библиотека IPC Tools интегрирована в программное обеспечение VoIP-платформы и обеспечивает передачу принятых из сети голосовых потоков от хост-процессора в DSP и обратно с помощью механизма логических межпроцессорных каналов, а также управление преобразованием голосовых потоков в DSP посредством механизма обмена сообщениями. Для каждого голосового потока создается отдельный логический канал. Требуемое количество контейнеров и размер связанных с ними буферов определяется приложением исходя из параметров потока (тип используемого голосового ко-

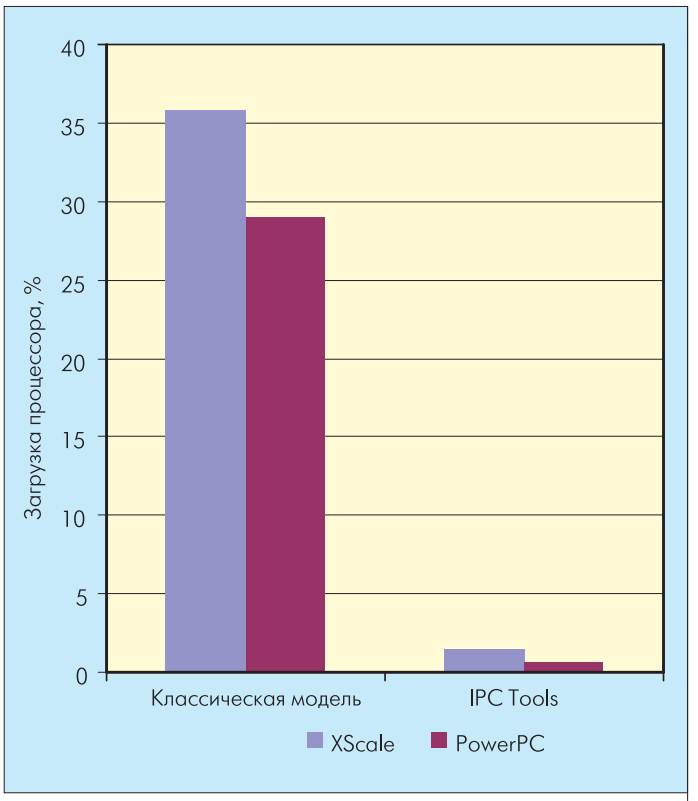

Рис.4. Прирост загрузки процессора (IBM PowerPC 405GPr и Intel XScale IXP425) в связи с операциями межпроцессорного обмена

дека, период формирования речевых пакетов и др.), согласованных с удаленным терминалом VoIP-сети через протоколы сигнализации (например, H.323, SIP и т.п.).

Периоды формирования пакетов для большинства голосовых соединений одинаковы (наиболее популярны значения 20 и 30 мс). Это позволяет привязать все каналы с одинаковыми периодами к одному таймеру. В обработчике таймера выполняются следующие действия:

- пакет изымается из очереди голосового соединения, упаковывается в контейнер и отправляется в логический межпроцессорный канал, ассоциированный с данным соединением;
- вызывается функция инициирования обмена.

Таким образом, в рамках одной межпроцессорной транзакции можно провести передачу сотен пакетов.

Механизм межпроцессорных сообщений IPC Tools используется также для управления дополнительной обработкой голосовых потоков, такой как распознавание и генерация тональной сигнализации.

Эффективность IPC Tools оценивалась в сравнении с классической моделью. Расчеты проводились для случая обработки 60 голосовых соединений (кодек G.711, дуплексный поток 64 кбит/с в каждом направлении на каждое соединение).

Применение IPC Tools позволило практически свести на нет программно-управляемый обмен через PCI, использовать DMA в цепочечном режиме и более чем на два порядка уменьшить количество обрабатываемых прерываний по сравнению с классической моделью.

В результате реализованных в IPC Tools оптимизаций удалось значительно снизить загрузку центрального процессора системы, связанную с ведением межпроцессорного обмена (рис. 4). В реальной работе это позволило увеличить производительность системы по сравнению с предыдущей версией библиотеки в 1,5–2,5 раза.

ЛИТЕРАТУРА

1. **Иванов В.Э., Койнов А.В.** Многопроцессорные системы беспроводной передачи данных: Материалы региональной научно-практической конференции "Методы разработки программного обеспечения". – http://webconf.rtf.ustu.ru/file.php/26/moddata/forum/48/242/ATT00009.doc

2. **Немнюгин С.А., Стесик О.Л.** Параллельное программирование для многопроцессорных систем. – СПб.: БХВ-Петербург, 2002.

3. **Таненбаум Э.** Современные операционные системы. 2-е изд. – СПб.: Питер, 2002.

4. **Иванов В.Э., Койнов А.В.** Исследование факторов, снижающих производительность многопроцессорных телекоммуникационных систем. – Телекоммуникации. – 2007, №5, с.19–23.