УДК 004.318 **DOI** 10.15622/ia.23.3.8

А.В. Сурченко, Ю.А. Недбайло

МЕТОДИКА КОМПРЕССИИ ДАННЫХ В НАКРИСТАЛЬНЫХ И МЕЖПРОЦЕССОРНЫХ СЕТЯХ С ШИРОКИМИ КАНАЛАМИ И ПОЛИТИКОЙ УПРАВЛЕНИЯ ПОТОКОМ WORMHOLE

Сурченко А.В., Недбайло Ю.А. **Методика компрессии данных в накристальных и межпроцессорных сетях с широкими каналами и политикой управления потоком wormhole.**

Аннотация. Увеличение количества вычислительных ядер является одним из основных современных способов повышения производительности процессоров. При этом увеличивается и нагрузка на подсистему памяти процессора в связи с растущим числом инициаторов обращений в память. Одним из нестандартных подходов к повышению производительности подсистемы памяти является аппаратная компрессия данных, позволяющая, во-первых, повысить эффективный объем кэш-памяти, снижая частоту запросов в оперативную память, а во-вторых, снизить интенсивность трафика в подсистеме памяти за счет более плотной упаковки данных. В работе рассматривается применение аппаратной компрессии данных в сети-на-кристалле и межпроцессорных каналах связи в конфигурации с широкими каналами передачи данных и политикой управления потоком wormhole. Существующие решения для такой конфигурации нельзя считать применимыми, т.к. они принципиально основаны на использовании узких каналов передачи данных и политиках управления потоком, предполагающих передачу пакета в неразрывном виде, что может не соблюдаться при применении политики wormhole. Предлагаемая в работе методика позволяет использовать аппаратную компрессию для рассматриваемой конфигурации за счет переноса процесса компрессии и декомпрессии из самой сети в соединяемые устройства, а также ряда оптимизаций по сокрытию задержек на преобразование данных. Рассматриваются оптимизации некоторых частных случаев передачи данных – передачи больших пакетов данных, состоящих из нескольких кэш-строк, а также нулевых данных. Особое внимание в работе уделено передаче данных по межпроцессорным каналам связи, в которых, в связи с их меньшей пропускной способностью по сравнению с сетью-на-кристалле, применение компрессии способно оказать наибольший эффект. Повышение пропускной способности подсистемы памяти при использовании в ней аппаратной компрессии данных подтверждается экспериментальными результатами, показывающими относительное увеличение IPC в задачах пакета SPEC CPU2017 до 14 процентов.

Ключевые слова: архитектура процессора, подсистема памяти, аппаратная компрессия данных, сеть-на-кристалле, межпроцессорные каналы связи, модель процессора.

1. Введение. В настоящее время одним из основных способов увеличения производительности процессоров общего назначения является увеличение числа входящих в них ядер. Этот подход позволяет увеличить количество задач или потоков, которые могут быть параллельно запущены на машине, как следствие, возрастает объем вычислений, который машина способна произвести за определенное время. Однако при этом возрастает нагрузка на подсистему памяти, поток запросов в которую увеличивается, а при

достаточно плотном потоке обращений ее пропускной способности может оказаться недостаточно. В связи с этим повышение пропускной способности подсистемы памяти становится одной из ключевых задач разработчиков современных процессорных систем [1 – 4].

Основные подходы к решению этой задачи включают в себя оптимизацию работы общей кэш-памяти для снижения частоты промахов в кэш и, как следствие, уменьшения частоты обращения в оперативную память, а также совершенствование топологии и архитектуры маршрутизаторов сети-на-кристалле процессора для повышения ее пропускной способности и снижения задержек на передачу.

Альтернативным подходом, уже достаточно длительное время обсуждаемым в научном сообществе, можно считать аппаратную компрессию данных. Ее применение позволяет преобразовать данные, как правило, на уровне кэш-строк, к меньшему размеру. За счет этого удается, с одной стороны, снизить трафик в подсистеме памяти [5 – 10], а с другой – увеличить эффективный объем кэш-памяти [11 – 13] или даже оперативной памяти [14 – 16] за счет того, что в памяти того же объема потенциально можно будет разместить большее количество кэш-строк.

Традиционные алгоритмы компрессии не являются подходящими для применения в аппаратуре микропроцессора, поскольку их использование может привести к появлению значительных дополнительных задержек по чтению данных, росту энергопотребления и нарушению структурной организации устройств. В связи с этим при создании алгоритмов, предназначенных для аппаратной компрессии данных, разработчики стараются учитывать эти факторы и минимизировать воздействие компрессии на параметры всей системы.

В данной работе рассматривается применение аппаратной компрессии при передаче данных, а именно в каналах сети-на-кристалле и в межпроцессорных каналах связи. Рассмотрим, как аппаратная компрессия данных может повлиять на факторы, которые ограничивают производительность этих каналов.

В отношении сети-на-кристалле необходимо рассмотреть ситуацию, когда в нее одновременно поступает большое число запросов от всех ядер системы. В таком случае в сети наблюдается повышенный трафик, и в ее узлах возрастает число конфликтных ситуаций, вызванных тем, что пакеты с нескольких входных направлений требуется передать в одном и том же выходном направлении. В ходе арбитража ресурс предоставляется только одному

из направлений, посылки с других направлений находятся в буфере вплоть до освобождения ресурса и их выбора арбитром. Именно это ожидание освобождения ресурса и является основной причиной ограничения производительности в сети-на-кристалле.

Использование аппаратной компрессии уменьшает размер пакета, в связи с чем уменьшается количество флитов – частей пакета, на которые он разбивается – а значит, передача одного пакета через узел занимает меньшее время. Как следствие, конфликтных ситуаций в сети становится меньше, а в тех случаях, когда они все же возникают, пакеты, проигравшие при арбитраже, будут проводить меньше времени ожидания в буфере. Таким образом, общая пропускная способность сети-на-кристалле возрастет.

Рассматривая межпроцессорные каналы связи, можно выделить несколько факторов, ограничивающих их пропускную способность. Во-первых, несмотря на порой более высокую частоту работы этих каналов, уже не привязанную к частоте работы процессора, их ширина ограничена количеством выводов процессора, выделенных под такую передачу. В связи с этим пропускная способность этих каналов в целом всегда ниже, чем сети-на-кристалле. Во-вторых, помимо пакетов с данными, по этим каналам передаются пакеты и других типов, соответствующих сообщениям системного протокола, связанным с поддержкой когерентности. Помимо пакетов системного протокола когерентности, по каналам могут также передаваться служебные сообщения, связанные с поддержкой помехоустойчивости, или необходимые для корректной обработки принимаемых данных. Однако за счет своих размеров пакеты с данными все еще будут составлять достаточно существенную долю сетевого трафика.

Из-за более низкого темпа передачи при высокой нагрузке на межпроцессорные каналы связи большая часть запросов будет помещена в буфер в ожидании передачи. Применение аппаратной компрессии позволит сократить объем передаваемых по каналам пакетов с данными, а значит, будет освобождено место для отправки дополнительных сообщений. За счет этого степень буферизации снизится, а пропускная способность межпроцессорных каналов связи возрастет [17].

2. Современное состояние. Актуальность исследования. Научные работы, связанные с аппаратной компрессией данных при их передаче, можно условно разделить на две группы – работы, посвященные используемым алгоритмам компрессии, и работы, связанные непосредственно с разработкой методик по осуществлению компрессии и взаимодействию со сжатыми данными.

В ходе рассмотрения алгоритмов компрессии в [18] они были разделены на несколько групп в зависимости от того, какой тип локальности данных используется алгоритмом для достижения компрессии. Авторы выделяют следующие типы локальности:

– локальность малых значений: отбрасываются нулевые или единичные разряды для малых положительных или отрицательных значений;

– локальность сгруппированных значений: используется слабое различие в значениях между находящимися по соседству данными;

– локальность изолированных значений: применяется кодирование часто встречающихся значений или последовательностей значений меньшим числом битов.

В одной из более поздних работ, посвященных алгоритмам компрессии, большое внимание уделено такому параметру алгоритма, как задержка декомпрессии [11]. Поскольку именно декомпрессия данных при работе алгоритма находится на критическом пути чтения данных, при разработке алгоритма необходимо добиться максимального сокращения затрачиваемого на нее времени.

Авторы статьи [11] предлагают свой алгоритм BΔI (Base-Delta-Immediate), основанный на локальности сгруппированных значений и работающий на уровне кэш-строк. Исходная строка разбивается на сегменты равного размера, и затем значения этих сегментов сравниваются с некоторым базовым значением. В том случае, если разница в значениях для каждого из сегментов достаточно мала, в сжатой кэш-строке размещается базовое значение и набор разностей для восстановления исходных значений сегментов. За счет того, что исходные значения восстанавливаются параллельно для каждого из сегментов, скорость декомпрессии при использовании этого алгоритма очень высока. В соответствии с проведенными расчетами, при аппаратной реализации декомпрессия по алгоритму BΔI укладывается в 1 такт при рабочей частоте 2 ГГц [18].

В данной работе используется модифицированная версия алгоритма, названная BΔI*-HL [19, 20]. Ее особенностями является модифицированное вычисление смещений в значениях между сегментами для ускорения компрессии и декомпрессии, а также упрощенная грануляция сжатых кэш-строк по конечным размерам с точностью до половины кэш-строки для лучшей адаптированности к аппаратной реализации. Как компрессия, так и декомпрессия по алгоритму BΔI*-HL укладываются в 1 такт при рабочей частоте 2 ГГц.

При выборе алгоритма для применения компрессии к передаваемым данным в последнее время наблюдается тенденция к рассмотрению алгоритмов на основе локальности сгруппированных значений, в то время как в более ранних работах преимущественно рассматривались алгоритмы на основе локальности изолированных значений [5]. Такой переход можно связать с тем, что при применении алгоритмов на основе локальности изолированных значений, как правило, для хранения таких значений используются справочники, которые могут формироваться как статически, так и динамически, на основе информации, передаваемой в ходе работы системы. С переходом от многоядерных процессоров на общей шине к реализации сетей-на-кристалле, сложность поддержки таких справочников существенно возрастает, т.к. придется содержать отдельные справочники для каждой пары взаимодействующих в сети-устройств или даже для каждого направления передачи в этой паре. Кроме того, потребуется поддерживать консистентность справочников у приемника или передатчика. В случае использования алгоритмов на основе локальности сгруппированных значений подобных проблем не возникает, т.к. вся информация, необходимая для восстановления исходных данных, уже содержится в самих сжатых данных.

Рассмотрим теперь последние работы, посвященные разработке методик по осуществлению компрессии передаваемых данных [5 – 10].

Подход NoΔ, рассматриваемый в [5], предполагает использование BΔI-подобного алгоритма в сетевых адаптерах устройств сети-на-кристалле для компрессии и декомпрессии. Рассматривая передачу флитов малой гранулярности (4 байта), авторы работы отмечают, что их подход позволяет довольно существенно снизить количество флитов, требуемое для передачи сжатых данных. Задержки на компрессию и декомпрессию при этом удается скрыть за счет того, что данные поступают на шину передачи в течение нескольких тактов. Оценка, полученная в работе, показывает, что рассматриваемый подход является более эффективным при использовании в сети-на-кристалле, чем компрессия нулевых или часто встречающихся значений.

В статье о FlitZip [6] предлагается дополнительно уменьшить число передаваемых флитов за счет частичного переноса метаданных компрессии в заголовочный флит, где часто остается неиспользованное пространство.

Авторы подхода DISCO [7, 8] выделяют дополнительную задержку на компрессию и декомпрессию, возникающую на критическом пути доступа в кэш, как одну из ключевых проблем,

сдерживающих применение аппаратной компрессии данных. В качестве решения они предлагают перенести процесс компрессии и декомпрессии данных в коммутаторы сети-на-кристалле. Возможность осуществления преобразования данных в предлагаемой ими методике определяется на основе занятости коммутаторов – компрессию или декомпрессию можно осуществить в том случае, если выдача данных в данный момент невозможна, и они находятся в буфере, ожидая разрешения на передачу. Помимо этого критерия, на решение влияет также ряд других эвристик, связанных с движением данных по сети.

Важно отметить, что все современные подходы основываются на определенной конфигурации сети – в ней должны быть сравнительно узкие каналы передачи, а также пакеты должны идти по каналам без разрывов между флитами. Требование об узких каналах передачи необходимо для того, чтобы задержка на преобразование данных была скомпенсирована задержкой на передачу одного пакета данных через определенную точку сети. Например, в случае NoΔ это задержка на выдачу или прием пакета в сетевом адаптере, а в случае DISCO – задержка из-за выдачи через коммутатор другого пакета, блокирующего текущий пакет, над которым осуществляется преобразование. Требование о неразрывной передаче пакетов необходимо для возможности их компрессии и декомпрессии без дополнительной задержки на ожидание всех частей пакета. Такому требованию удовлетворяют политики управления потоком store-and-forward и virtual cut-through, но для политики wormhole это требование может не соблюдаться. При работе этой политики выдача кредитов для передачи сообщений происходит на уровне флитов, а не самих пакетов. Как следствие, возможна ситуация, когда кредиты заканчиваются в момент, когда переданы еще не все флиты сообщения. Часть флитов в этом случае передается дальше, а другая остается в буфере, таким образом возникает разрыв. Авторы подхода DISCO уделяют этому внимание, предлагая компромиссный вариант с компрессией частей передаваемого пакета, состоящих из нескольких флитов, позволяющего продолжать осуществлять компрессию, хотя и с меньшей эффективностью.

Рассмотрим теперь конфигурацию сети, где используются каналы передачи большой ширины с политикой управления потоком wormhole, что не рассматривалось авторами описанных выше работ. В этом случае задержки на передачу пакета оказывается недостаточно для полноценной компенсации задержки декомпрессии, а компромиссный подход по компрессии частей пакета, предложенный

авторами DISCO, перестает работать в связи с тем, что при широких каналах передачи пакет разбивается на существенно меньшее число флитов. В связи с этим актуальной становится разработка новой методики, адаптированной для данной конфигурации сети.

3. Цель исследования и постановка задачи. Целью данного научного исследования является разработка методики, позволяющей применять аппаратную компрессию данных в накристальных и межпроцессорных сетях с широкими каналами передачи данных и политикой управления потоков wormhole.

Научная новизна полученных в работе результатов определяется разработкой новой методики применения аппаратной компрессии в процессорных каналах передачи данных, предназначенной для каналов передачи большой ширины, а также разработкой ряда оптимизаций частных случаев передачи данных за счет аппаратной компрессии.

Теоретическая значимость работы заключается в том, что разработанная в ее рамках методика позволяет применять аппаратную компрессию данных в процессорных каналах передачи большой ширины, при этом как в накристальных сетях, так и в межпроцессорных каналах связи. Практическая значимость работы заключается в возможности использования разработанной методики для повышения пропускной способности широких каналов передачи данных в процессорах общего назначения за счет аппаратной компрессии.

Общая постановка задачи исследования сводится к рассмотрению типичной конфигурации сети с широкими каналами передачи и выявлению ее особенностей по сравнению с конфигурациями, предназначенными для узких каналов, а затем к разработке методики для поддержки в такой конфигурации аппаратной компрессии, учитывающей выявленные особенности и позволяющей обрабатывать сжатые данные как в накристальных сетях, так и в межпроцессорных каналах связи.

Эффективность применения методики оценивается на основе изменения показателей интенсивности трафика и IPC (Instructions per Clock, количества инструкций, исполняемых за процессорный такт) при применении аппаратной компрессии на основе рассматриваемой методики в модели процессора архитектуры «Эльбрус».

В качестве ограничения по применению методики рассматриваются только такие конфигурации сети, которым свойственны определенные особенности в передаче пакетов, связанные с большей шириной каналов. При этом типичный пакет с данными

в таких конфигурациях должен иметь размер, больший, чем один флит, в противном случае аппаратная компрессия не будет оказывать положительного эффекта на пропускную способность.

К допущениям при рассмотрении методики можно отнести выровненность массивов данных и их организацию в порядке байт Little Endian, а также малую задержку на компрессию и декомпрессию данных (достижимую при использовании алгоритма компрессии BΔI*-HL).

Областью применения методики является подсистема памяти современных процессоров общего назначения.

4. Рассматриваемая конфигурация. В качестве примера конфигурации с широкими каналами передачи и политикой управления потоком wormhole, на основе которой будет разрабатываться методика, была выбрана конфигурация сети, используемая в процессорах архитектуры «Эльбрус».

Определяющей особенностью рассматриваемой конфигурации является использование широких каналов передачи данных, выбор которых связан с необходимостью обеспечить высокий темп передачи данных процессорным ядрам. Эта характеристика напрямую связана с шириной каналов, т.к. соответствует объему данных, получаемому ядром в единицу времени. Она особенно критична из-за отсутствия в ядрах поддержки внеочередного исполнения инструкций и простаивания арифметических устройств в ожидании операндов.

Большая ширина каналов также позволяет сократить время, необходимое для выдачи или приема пакета с данными в сетевых каналах. В тактах это время соответствует числу флитов, на которые делится пакет. Передача данных одного и того же размера потребует в конфигурации с широкими каналами меньшего числа флитов, чем в конфигурации с узкими каналами.

В конфигурации сети, используемой в процессорах архитектуры «Эльбрус», типичный пакет данных, в котором передается одна кэш-строка, состоит из двух флитов, в каждом из которых передается одна из половин кэш-строки. Пакет дополняется заголовком, содержащим информацию для маршрутизации и идентификации пакета. С целью повышения пропускной способности сети для заголовков выделены отдельные линии передачи, так что в сети возможна одновременная передача одного заголовка и одного флита самого пакета.

Поскольку для заголовков в сети выделены отдельные линии, они передаются с каждым флитом пакета, т.е. с пакетом передается не один заголовок, а несколько, равное количеству флитов пакета. Подобный подход не приводит к дополнительным накладным

расходам, зато позволяет осуществлять маршрутизацию в случае разрывов между флитами одного пакета, возможных при политике управления потоком wormhole. В этом случае кредитный механизм, на основе которого работает передача между узлами сети, функционирует на уровне отдельных флитов, поэтому возможны ситуации, когда кредитов хватило для передачи только части пакета. Другой ситуацией, которая может привести к разрывам между флитами пакета, является пересинхронизация на границе доменов, регулируемых разными синхросигналами.

Особенностью передачи пакетов с данными в рассматриваемой конфигурации, по сравнению с пакетами других типов, является передача заголовков пакета на такт раньше, чем соответствующих им флитов. Это связано с тем, что за счет большей ширины каналов мультиплексирование данных начинает занимать большее время, которое сложно будет уложить в 1 такт. Передача заголовка на такт раньше позволяет заранее определить, откуда поступают данные, а затем провести мультиплексирование данных на следующем такте.

Схематичный внешний вид пакета с данными в рассматриваемой конфигурации, в котором передается одна кэш-строка, показан на рисунке 1. Пакет состоит из двух флитов, заголовки для которых передаются на такт раньше. Дополнительно вместе с заголовками передается сигнал last, обозначающий последний флит пакета.

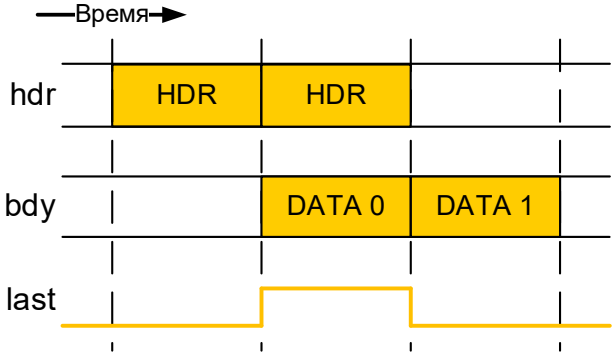

Рис. 1. Временная диаграмма передачи пакета с данными, в котором содержится одна кэш-строка

Еще одной особенностью рассматриваемой конфигурации сети является тот факт, что разбиение данных на отдельные флиты в ней производится не в сетевых адаптерах, а в самих устройствах,

до поступления данных в сеть-на-кристалле. На это есть несколько причин. Во-первых, данные для передачи могут быть готовы раньше, чем такая передача будет разрешена по протоколу когерентности. В связи с этим появляется возможность предварительно упорядочить данные в том виде, в котором они должны поступить в сеть. Во-вторых, чтение данных из памяти устройств производится, как правило, по частям, занимая несколько тактов. Поэтому, даже если данные еще не прочитаны полностью, благодаря их последовательному чтению по частям они могут быть направлены в сетевой адаптер в последовательном порядке сразу же после прочтения, по одному флиту в такт.

Межпроцессорные каналы связи подключаются к сети-на-кристалле процессора за счет контроллеров межпроцессорного обмена в количестве, равном числу процессоров, с которым будет взаимодействовать данный процессор. Пакеты, поступающие в контроллер из сети-на-кристалле, попадают в буфер, а затем производится их арбитраж для упаковки в контейнеры, являющиеся единицей передачи информации в межпроцессорных каналах связи. Данный арбитраж зависит от типа пакета (например, наивысший приоритет отдается сообщениям для завершения транзакций, а наименьший – первичным запросам, инициирующим транзакцию), их размеров и времени нахождении в буфере (чтобы, несмотря на низкий приоритет, все пакеты могли быть отправлены при высокой нагрузке на канал).

Описанные выше особенности рассматриваемой конфигурации, необходимые для работы с широкими каналами передачи, в целом слабо зависят от прочих параметров сети. Это позволяет прийти к выводу о том, что конфигурации сети, адаптированные для работы с широкими каналами передачи, которые используются в процессорах других архитектур, слабо отличаются от рассматриваемой. Таким образом, предлагаемая в работе методика может быть применена для любых конфигураций сети с широкими каналами передачи.

5. Предлагаемая методика. Одним из основных вопросов, которые требуется рассмотреть при интеграции аппаратной компрессии данных в процесс передачи данных, является определение точек, в которых будет производиться компрессия и декомпрессия данных. В подходах NoΔ и FlitZip, предложенных в [5] и [6], преобразование данных производится в сетевых адаптерах устройств, в подходе DISCO [7, 8] – в узлах сети в случаях задержки передачи данных.

При использовании рассматриваемой конфигурации сети перечисленные выше подходы не применимы. Это связано с тем, что, как в сетевых адаптерах, так и в самой сети, данные уже представлены в виде нескольких флитов. Для осуществления компрессии потребуется дождаться получения обоих флитов данных, а также поместить информацию о компрессии в заголовок пакета. С учетом задержки на компрессию в 1 такт, суммарная задержка заняла бы 3 такта, как показано на рисунке 2. Столь длительная дополнительная задержка неприемлема и не позволяет осуществлять компрессию в сетевых адаптерах или самой сети.

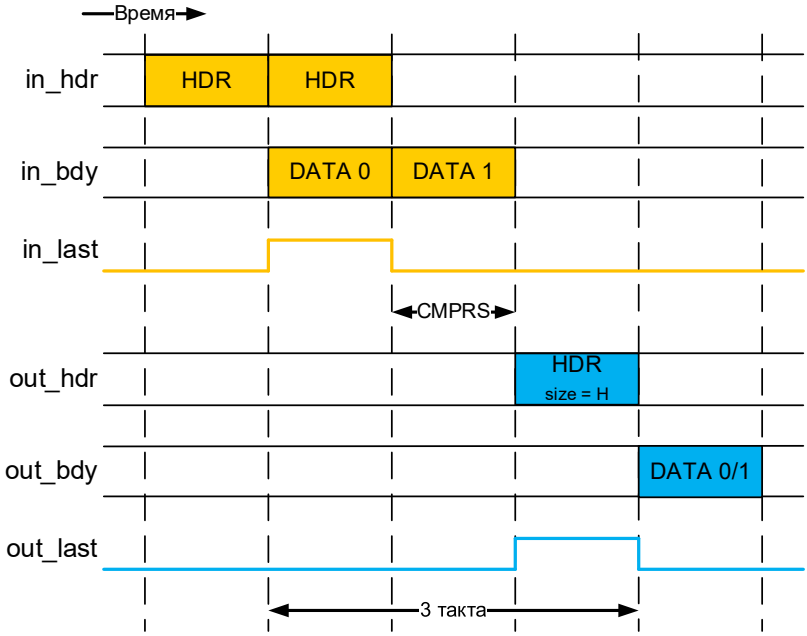

Рис. 2. Задержка при компрессии данных в сетевых адаптерах или самой сети-на-кристалле

При декомпрессии в сетевых адаптерах или сети дополнительной задержки, помимо задержки в 1 такт на декомпрессию, не возникнет. Во-первых, преобразование заголовка, для того чтобы убрать оттуда информацию о компрессии, можно произвести, не дожидаясь данных, а во-вторых, сжатые данные занимают только один флит, поэтому после его получения можно сразу же приступать к декомпрессии. Схематично процесс декомпрессии

представлен на рисунке 3. Единственной ситуацией, когда может потребоваться дополнительная задержка, является последовательная передача двух пакетов с данными, первый из которых является сжатым. В таком случае перед приемом второго пакета придется вставить однотактовую задержку в связи с расширением первого пакета до двух флитов в ходе декомпрессии.

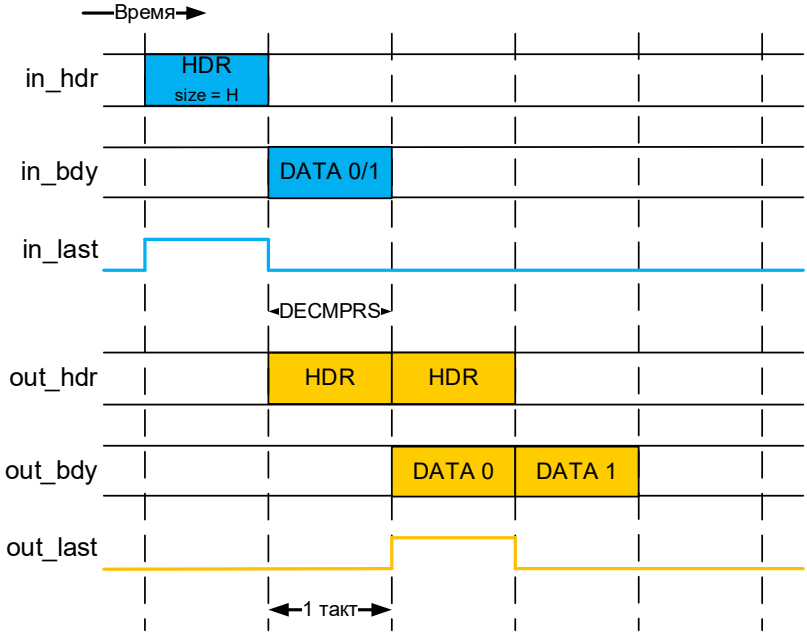

Рис. 3. Задержка при декомпрессии данных в сетевых адаптерах или самой сети-на-кристалле

Альтернативными точками применения компрессии и декомпрессии можно выбрать сами устройства, осуществляющие выдачу и прием данных. В таком случае преобразование данных будет осуществляться до передачи их в сеть, так что в сетевых адаптерах или сети данные будут передаваться в уже сжатом виде.

Такой подход особенно хорошо подходит для работы с данными записи в процессорах «Эльбрус». В протоколе когерентности, используемом в этих процессорах, записи являются непочтовыми, т.е. для отправки данных необходимо получить разрешение от получателя о готовности к приему. Интервал между готовностью данных и получением разрешения на их отправку может

использоваться для осуществления компрессии, как показано на рисунке 4. Дополнительных задержек передачи из-за компрессии при этом не возникает. При приеме данных все еще возникнет задержка на их декомпрессию. В прочих конфигурациях, где записи могут являться почтовыми, т.е. не требующими разрешения на отправку, при передаче данных будет проявляться задержка как на их компрессию, так и на декомпрессию, но, поскольку запись данных не находится на критическом пути, эта задержка не окажет существенного влияния на производительность.

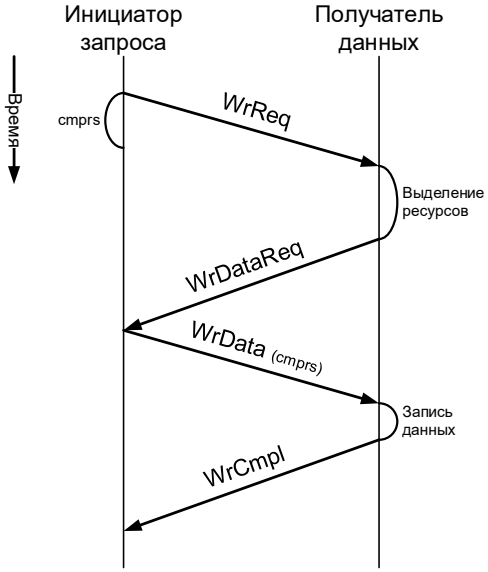

Рис. 4. Компрессия данных в ходе транзакции по записи данных

При работе с данными чтения в общем случае устранить задержки невозможно, т.к., в отличие от записей, данные чтения могут быть сразу отправлены инициатору запроса, поскольку при инициировании запроса на чтение он сразу подготавливается к приему данных. Задержки могут быть скрыты частично, если разрешить устройствам, которые не модифицируют данные, таким как общая кэш-память или оперативная память, хранить данные в сжатом виде. В таком случае запись этих данных не потребует декомпрессии, а при чтении не потребуется компрессия данных, нужна будет только их декомпрессия на принимающей стороне.

Предлагаемый подход можно сочетать с методиками по компрессии данных в кэш-памяти [19] или оперативной памяти. В таком случае сжатые данные не нужно подвергать декомпрессии перед выдачей из устройства.

При передаче данных через межпроцессорные каналы связи появляется дополнительная возможность компрессии данных, связанная с их достаточно длительной буферизацией. Компрессия позволит уменьшить объем передаваемых данных минимум вдвое, а значит, появится возможность добавить в освободившееся место в контейнере другие пакеты.

Для поддержки работы со сжатыми данными в контейнерах должны поддерживаться распределения пакетов по контейнеру, где данные имеют только половинный размер. Однако в рассматриваемой конфигурации подобные пакеты уже используются – они применяются для передачи данных чтения в том случае, когда считывается только половина кэш-строки. В связи с этим переработка распределений пакетов по контейнерам в рассматриваемой конфигурации не потребуется.

6. Дополнительные оптимизации частных случаев передачи данных. Применяя аппаратную компрессию к сети-на-кристалле и межпроцессорным каналам связи процессора, можно дополнительно оптимизировать несколько частных случаев передачи данных.

Одним из таких случаев является передача пакетов с данными большого размера (в рассматриваемой конфигурации – вплоть до 4 кэш-строк). Такие данные могут передаваться при работе механизма Peer-to-Peer в режиме прямого доступа к памяти (Direct Memory Access, или DMA), когда два внешних по отношению к процессору устройства могут обмениваться данными напрямую без использования оперативной памяти процессора.

Использование компрессии для больших пакетов с данными может существенно сократить их размер. Как и для обычных пакетов, компрессия будет производиться на уровне одной кэш-строки, однако теперь, помимо информации, необходимой для корректной декомпрессии, вместе с каждым заголовком будет передаваться маска, показывающая, была ли определенная строка из передаваемых сжата до нулевого размера. В этом случае флиты, относящиеся к этой строке, можно не передавать. Пример подобной передачи приведен на рисунке 5, где передается большой пакет из 4-х кэш-строк. Кэш-строки 0 и 3 сжимаются до половинного размера, кэш-строка 1 не сжимается, а кэш-строка 2 сжимается до нулевого размера и не передается.

Информация о том, что кэш-строка 2 «пропущена» в передаваемых данных, отражена в передаваемой маске.

На рисунке 6 показан частный случай передачи большого пакета с данными, когда все 4 передаваемые кэш-строки нулевые. В таком случае вместо 8 флитов достаточно будет передать только 1, в котором по маске на принимающей стороне можно будет понять, что нужно восстанавливать нулевые данные.

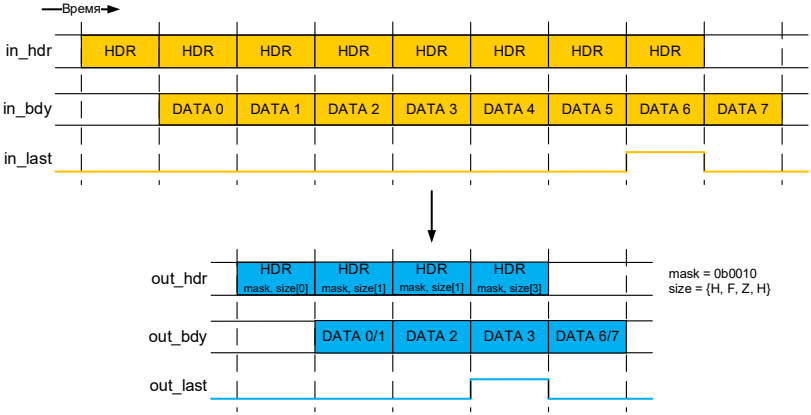

Рис. 5. Компрессия большого пакета с данными из 4 кэш-строк

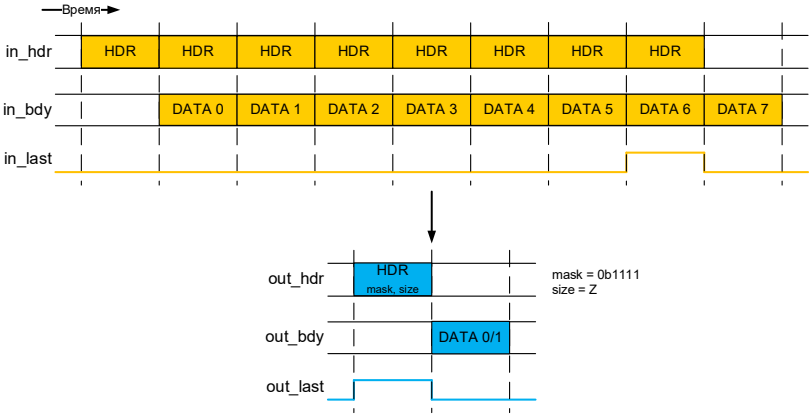

Рис. 6. Компрессия большого пакета с данными из 4 кэш-строк в случае, когда все данные нулевые

Как и при передаче пакетов стандартного размера, наиболее оптимально применение такой компрессии для данных записи, однако

в случае работы с данными чтения можно частично скрыть задержку на компрессию и декомпрессию за счет того, что число передаваемых флитов может существенно уменьшиться.

Другим частным случаем, который можно оптимизировать, является передача нулевых данных. В этом случае пакет с данными можно было бы не передавать, обходясь только передачей одного бита информации о нулевом значении данных. Для этого необходимо определить, каким именно образом эта информация может передаваться.

В случае запросов по записи информация о значении данных имеется у инициатора запроса, отправляющего данные. В связи с этим информация о нулевом значении данных может направляться сразу же вместе с первичным запросом по записи. Получатель запроса в таком случае не должен высылать подтверждение приема данных, а подготовиться к записи и самостоятельно записать нулевые значения, после чего выдать сигнал завершения транзакции. Данный порядок действий проиллюстрирован на рисунке 7.

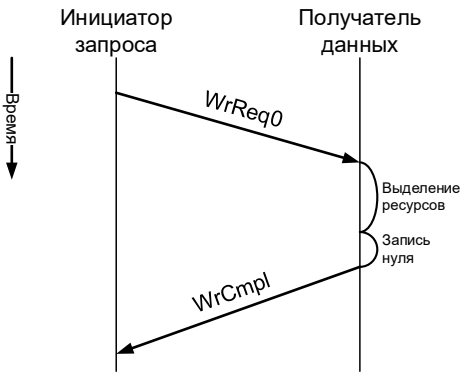

Рис. 7. Транзакция по записи нулевых данных

При запросах на чтение информация о значении данных имеется у их владельца, поэтому их потребуется передавать инициатору запроса вместе с сообщениями-ответами на первичные запросы по чтению или на снуп-запросы (запросы по чтению, поступающие от справочника устройствам, обладающим копией данных). Необходимо также учитывать, что из-за особенностей протокола когерентности устройство-владелец данных может выдавать ответы не напрямую устройству-инициатору запроса, и в таком случае для передачи информации о нулевых данных потребуется использовать

несколько сообщений-ответов. Так, на рисунке 8 показана передача ответа напрямую устройству-инициатору запроса, а на рисунке 9 взаимодействие происходит через справочник, в связи с чем ответ с признаком нулевых данных сначала отправляется также в справочник, а затем уже из справочника передается устройству-инициатору запроса.

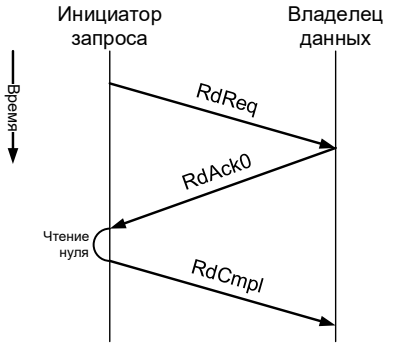

Рис. 8. Транзакция по чтению нулевых данных, передача информации о нулевом значении данных напрямую инициатору запроса

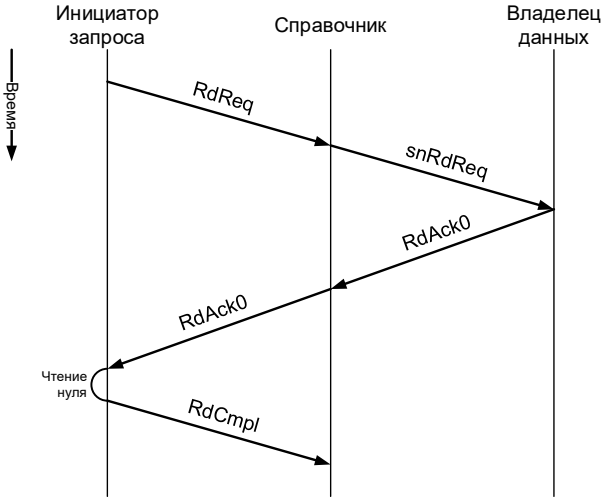

Рис. 9. Транзакция по чтению нулевых данных, передача информации о нулевом значении данных через справочник

7. Экспериментальные результаты. Исследование эффективности применения аппаратной компрессии при передаче данных по подсистеме памяти процессора производилось с помощью модели процессора архитектуры «Эльбрус» на основе трасс событий, изображенной на рисунке 10 и описанной в [21]. Моделируемая конфигурация соответствует процессору Эльбрус-16С и приведена в таблице 1.

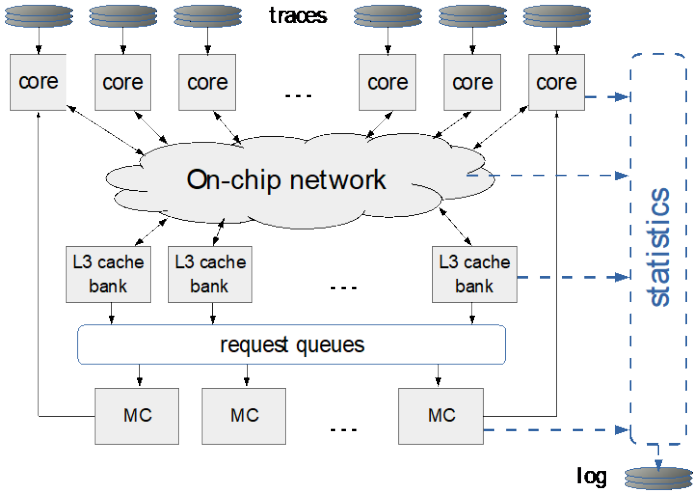

Рис. 10. Модель процессора «Эльбрус» на основе трасс событий

Таблица 1. Конфигурация процессора, используемая в симуляции

Компонент	Конфигурация
Ядро	16 VLIW-ядер, частота 2 ГГц
L1I-кэш	Частный, 128 КБ/ядро, 4-way, кэш-строка 256 Б
L1D-кэш	Частный, 64 КБ/ядро, 4-way, кэш-строка 32 Б, write-through
L2-кэш	Частный, 1 МБ/ядро, 4-way, кэш-строка 64 Б, неинклюзивный
L3-кэш	Общий, 2 МБ/ядро, 16-way, кэш-строка 64 Б, NCID
Сеть-на-кристалле	4x4 mesh, 1 запрос + 32 Б данных на один такт ядра
Оперативная память	8 каналов, DDR4 с частотой 3200 MT/s

На рисунке 11 продемонстрирована интенсивность трафика данных в пересчете на одно ядро и один такт в сети-на-кристалле без компрессии, вызванного промахами в L2 (столбцы OCN) и L3 (столбцы

MC) для задач пакета SPEC CPU2017 при описанной в таблице 1 конфигурации процессора. Рисунок 12 показывает долю сжатых кэш-строк, достижимую при работе алгоритма компрессии в сети-на-кристалле (OCN) и межпроцессорных каналах связи (IPCC) соответственно.

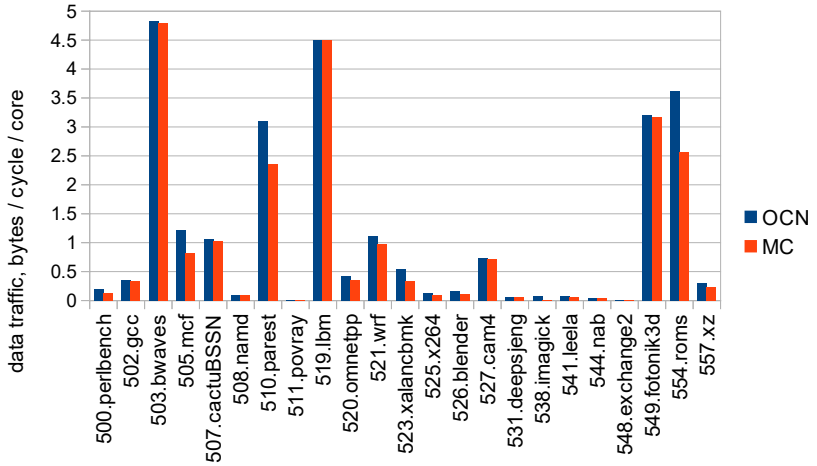

Рис. 11. Объем трафика данных сети-на-кристалле, вызванного промахами в L2 и L3

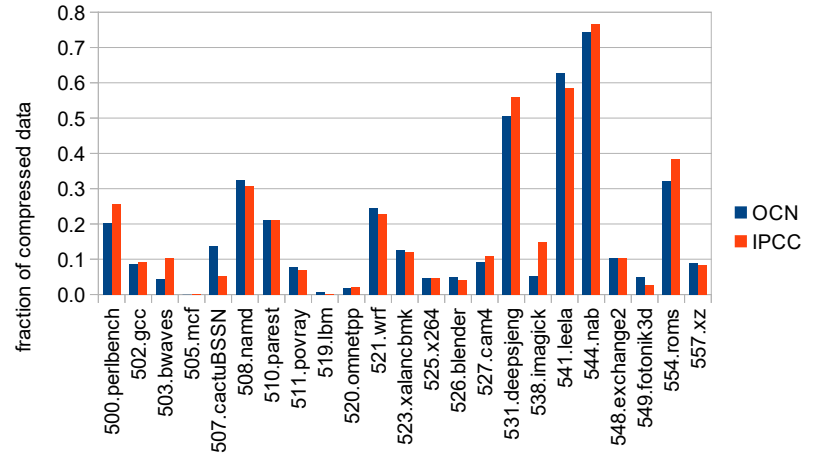

Рис. 12. Доля сжатых строк при применении алгоритма компрессии в сети-на-кристалле и межпроцессорных каналах связи

Эффективность компрессии данных в сети-на-кристалле моделировалась с ограничением пропускной способности этой сети по данным до 4 байт за такт на ядро, что соответствует одному 32-байтному флиту за 8 тактов для случаев большего количества ядер, другой топологии или/и меньшей частоты сети относительно ядер. Относительное увеличение IPC (Instructions per Clock, количество инструкций, исполняемых за процессорный такт) за счет использования аппаратной компрессии данных в тестах пакета SPEC CPU2017 при такой конфигурации, а также среднее геометрическое этого значения по всем задачам, представлено на рисунке 13. Наибольший прирост (вплоть до 7,1%) наблюдается в задачах 554.roms, 503.bwaves и 507.cactuBSSN, где в сети одновременно наблюдается высокий трафик данных (из-за частых промахов в L2 кэш) и высокая доля сжатых данных. В процессоре с меньшим размером L2 кэша или менее оптимизированных задачах эффект от аппаратной компрессии данных может быть ещё больше.

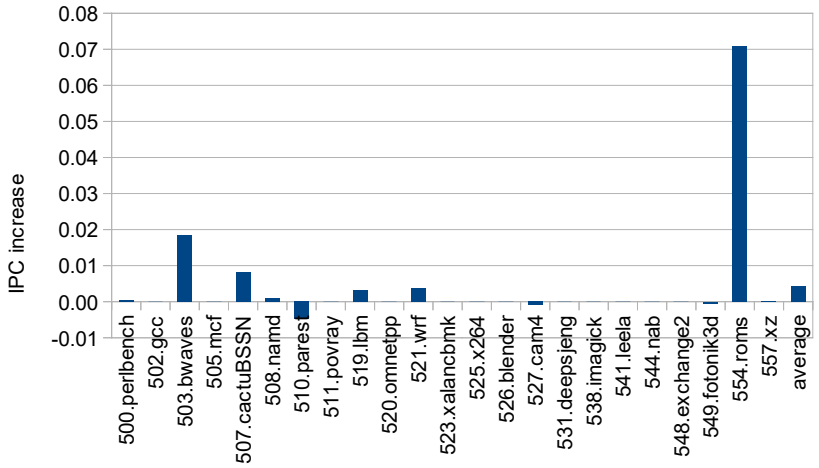

Рис. 13. Относительное увеличение IPC для задач пакета SPEC CPU2017 при аппаратной компрессии в сети-на-кристалле

Для моделирования применения аппаратной компрессии данных в межпроцессорных каналах связи пропускная способность доступа к некоторым страницам памяти (при их размере в 2 МБ) была ограничена величиной 10 ГБ/с с добавлением задержки в 200 нс. Передача сжатого пакета данных при этом составляла 70 байт, а несжатого – 103 байт суммарного трафика по каналу. Относительное

увеличение IPC благодаря компрессии в зависимости от доли таких страниц на такой модели показано на рисунке 14. Приведены результаты, полученные на задачах пакета SPEC CPU2017, а также среднее геометрическое по всем задачам.

С увеличением доли страниц, располагающихся в чужом процессоре, увеличивается среднее время доступа в память, из-за чего в некоторых задачах уменьшается трафик; соответственно, эффект от оптимизации зависит от доли таких страниц по-разному для разных задач. В среднем наибольший относительный прирост IPC при использовании компрессии наблюдается при 50% страниц в памяти чужого процессора, составляя 1,4%. Максимальный прирост, равный 14,0%, наблюдается в тесте 554.roms при 25% страниц в памяти чужого процессора.

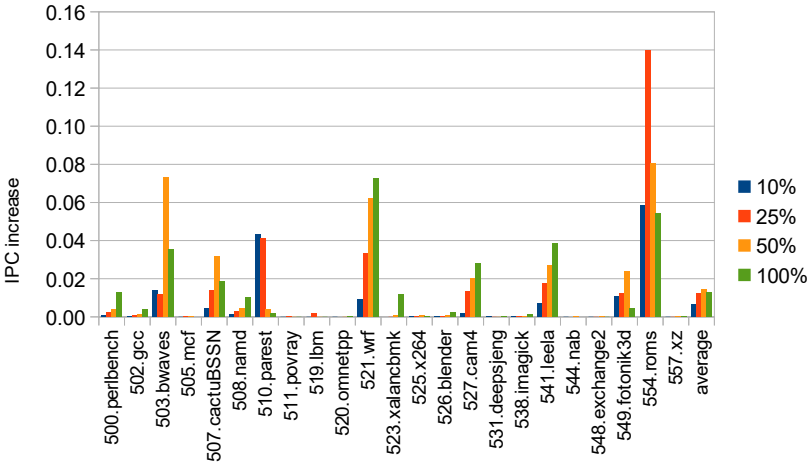

Рис. 14. Относительное увеличение IPC для задач пакета SPEC CPU2017 с аппаратной компрессией в межпроцессорных каналах связи при различных долях страниц памяти в чужих процессорах

8. Выводы. В рамках данной работы была достигнута цель научного исследования, заключающаяся в разработке новой методики применения аппаратной компрессии передаваемых данных в подсистеме памяти процессоров общего назначения, позволяющей использовать ее во внутрипроцессорных и межпроцессорных сетях с широкими каналами передачи данных и политикой управления потоком wormhole.

Применение новой методики в сети-на-кристалле предполагает перенос компрессии и декомпрессии данных в устройства сети. За счет этого задержки на преобразование данных в ходе компрессии или декомпрессии могут быть частично скрыты для запросов по чтению данных и полностью скрыты для запросов по записи данных.

В межпроцессорных каналах связи в рамках новой методики компрессию данных предлагается осуществлять без задержек, пользуясь фактом буферизации данных в контроллере межпроцессорных каналов перед их отправкой.

Дополнительно в рамках исследования разрабатываются оптимизации передачи пакетов с данными большого размера и передачи нулевых данных. Передача пакетов большого размера оптимизируется за счет передачи в заголовке маски нулевых кэш-строк, позволяющей не отправлять флиты большого пакета, относящиеся к нулевым кэш-строкам. Передачу нулевых данных предлагается оптимизировать за счет того, что вместо пакета с нулевыми данными вместе с другими сообщениями протокола когерентности будет передаваться признак нулевых данных.

Эффективность рассматриваемой методики подтверждается результатами экспериментов, в рамках которых оценивается применение аппаратной компрессии данных на основе методики к сети-на-кристалле и межпроцессорным каналам связи процессора в ходе работы задач пакета SPEC CPU2017.

В сети-на-кристалле, модифицированной для повышения нагрузки на нее, на некоторых задачах SPEC CPU2017 наблюдается относительное увеличение IPC до 7,1%. Однако при этом на сравнительно большом количестве задач относительного прироста IPC не наблюдается, что, согласно проведенным измерениям, можно связать либо с малым объемом трафика в сети-на-кристалле в этих задачах (т.е. с малым числом промахов в L2-кэш), либо с плохой сжимаемостью данных в задаче. Анализируя полученные результаты, можно сделать вывод о том, что задачи, в которых одновременно будет достаточно большой трафик в сети и высокая доля сжатых строк, получили бы наибольшее ускорение за счет аппаратной компрессии.

При моделировании использования аппаратной компрессии в межпроцессорных каналах связи было достигнуто относительное увеличение IPC вплоть до 14,0%. В целом максимальная степень относительного увеличения IPC для конкретных задач достигается при разных долях страниц в других процессорах, что показывает, что в определенных случаях положительный эффект от применения

компрессии может ограничиваться общей задержкой на передачу данных по межпроцессорным каналам связи.

Результаты работы позволяют применять аппаратную компрессию данных в сетях с широкими каналами передачи и политикой управления потоком wormhole, а также подтверждают, что аппаратная компрессия данных является достаточно эффективным механизмом повышения производительности подсистемы памяти.

Литература

1. Serpa M.S., Moreira F.B., Navaux P.O., Cruz E.H., Diener M., Griebler D., Fernandes L.G. Memory performance and bottlenecks in multicore and GPU architectures. 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). IEEE, 2019. pp. 233–236.
2. Mohamed A.M., Mubark N., Zagloul S. Performance aware shared memory hierarchy model for multicore processors. Scientific Reports. 2023. vol. 13(1). no. 7313.
3. Iyer R., De V., Illikkal, R., Koufaty, D., Chitlur, B., Herdrich, A., Khellah M., Hamzaoglu F., Karl E. Advances in microprocessor cache architectures over the last 25 years. IEEE Micro. 2021. Т. 41. № 6. С. 78–88.
4. Papazian I.E. New 3rd Gen Intel® Xeon® Scalable Processor (Codename: Ice Lake-SP) // Hot Chips Symposium. 2020. С. 1–22.
5. Zhan J., Poremba M., Xu Y., Xie Y. NoΔ: Leveraging delta compression for end-to-end memory access in NoC based multicores. 19th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2014. pp. 586–591.
6. Deb D., Rohith M.K., Jose J. Flitzip: Effective packet compression for noc in multiprocessor system-on-chip // IEEE Transactions on Parallel and Distributed Systems. 2021. Т. 33. № 1. pp. 117–128.
7. Wang Y., Han Y., Zhou J., Li H., Li X. DISCO: A low overhead in-network data compressor for energy-efficient chip multi-processors // Proceedings of the 53rd Annual Design Automation Conference. 2016. С. 1–6.
8. Wang Y., Li H., Han Y., Li X. A low overhead in-network data compressor for the memory hierarchy of chip multiprocessors // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2017. vol. 37. no. 6. pp. 1265–1277.
9. Li X., Sondhi T. FlitReduce: Improving Memory Fabric Performance via End-to-End Network Packet Compression. UC Berkeley CS262A Report. 2021. 9 p.
10. Pullaiah T., Manjunathachari K., Malleswari B.L. BΔ-NIS: Performance analysis of an efficient data compression technique for on-chip communication network. Integration. 2023. vol. 89. pp. 83–93.
11. Pekhimenko G., Seshadri V., Mutlu O., Gibbons P.B., Kozuch M.A., Mowry T.C. Base-delta-immediate compression: Practical data compression for on-chip caches // Proceedings of the 21st international conference on Parallel architectures and compilation techniques. 2012. С. 377–388.
12. Gaur J., Alameldeen A.R., Subramoney S. Base-victim compression: An opportunistic cache compression architecture // ACM SIGARCH Computer Architecture News. 2016. vol. 44. no. 3. pp. 317–328.
13. Carvalho D.R., Seznec A. Understanding cache compression // ACM Transactions on Architecture and Code Optimization (TACO). 2021. vol. 18. no. 3. pp. 1–27.
14. Pekhimenko G., Seshadri V., Kim Y., Xin H., Mutlu O., Gibbons P.B., Kozuch M.A., Mowry T.C. Linearly compressed pages: A low-complexity, low-latency main memory compression framework // Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture. 2013. С. 172–184.

15. Young V., Kariyappa S., Qureshi M.K. CRAM: Efficient Hardware-Based Memory Compression for Bandwidth Enhancement // arXiv preprint arXiv:1807.07685. 2018.
16. Choukse E., Erez M., Alameldeen A.R. Compresso: Pragmatic main memory compression // 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2018. С. 546–558.
17. Сурченко А.В. Исследование применимости аппаратной компрессии данных в межпроцессорных каналах связи процессоров с архитектурой Эльбрус // Труды Института системного программирования РАН. 2022. Т. 34. № 1. С. 49–58.
18. Thuresson M., Spracklen L., Stenstrom P. Memory-link compression schemes: A value locality perspective // IEEE Transactions on Computers. 2008. vol. 57. no. 7. pp. 916–927.
19. Kozhin A.S., Surchenko A.V. Design of Data Compression Mechanism in Cache Memory of Elbrus Processors // International Conference Engineering and Telecommunication (En&T). IEEE, 2020. С. 1–5.
20. Nedbailo Y.A., Surchenko A.V., Bychkov I.N. Reducing miss rate in a non-inclusive cache with inclusive directory of a chip multiprocessor // Computer Research and Modeling. 2023. vol. 15. no. 3. pp. 639–656.
21. Nedbailo Y. Fast and scalable simulation framework for large in-order chip multiprocessors // 26th Conference of Open Innovations Association (FRUCT). IEEE, 2020. pp. 335–345.

Сурченко Александр Викторович — старший инженер, АО "МЦСТ"; преподаватель кафедры, кафедра информатики и вычислительной техники, МФТИ (НИУ). Область научных интересов: производительность и поддержка когерентности подсистемы памяти процессоров общего назначения, аппаратная компрессия данных, кэш-память. Число научных публикаций — 10. Alexander.V.Surchenko@mcst.ru; улица Вавилова, 24, 119049, Москва, Россия; р.т.: +7(499)135-8969.

Недбайло Юрий Александрович — канд. техн. наук, ведущий инженер, АО "МЦСТ"; ведущий инженер, ПАО «ИНЭУМ им. И.С. Брука». Область научных интересов: подсистема памяти процессоров общего назначения. Число научных публикаций — 25. yuri.nedbailo@mail.ru; улица Вавилова, 24, 119049, Москва, Россия; р.т.: +7(916)936-8670.

DOI 10.15622/ia.23.3.8

A. SURCHENKO, YU. NEDBAILO

HARDWARE COMPRESSION METHOD FOR ON-CHIP AND INTERPROCESSOR NETWORKS WITH WIDE CHANNELS AND WORMHOLE FLOW CONTROL POLICY

Surchenko A., Nedbailo Yu. **Hardware Compression Method for On-Chip and Interprocessor Networks with Wide Channels and Wormhole Flow Control Policy.**

Abstract. Increasing the number of processing cores is currently a common way to boost processor performance. However, the load on the memory subsystem consequently increases as the number of its agents grows. Hardware data compression is an unconventional approach to improving memory subsystem performance by reducing, firstly, the main memory access rate by increasing the cache capacity and, secondly, data traffic by packing the data more densely. The paper describes the implementation of hardware data compression in the on-chip network and interprocessor links of a configuration with wide data transmission channels and a wormhole flow control policy. The existing solutions cannot be applied to such configurations because they are essentially based on using narrow data channels and flow control policies implying uninterrupted packet transmission, which is not maintained with the wormhole flow control. The method proposed in this paper enables the use of hardware compression in the aforementioned configuration by moving data compression and decompression from networks to the connected devices, as well as by using a number of optimizations to hide the data processing delays. Optimizations of some specific cases, such as the transmission of large data packets with several cache lines or the transmission of zero data, are considered. Special attention is given to data transmission via interprocessor links, where, due to their lower bandwidth compared to the on-chip network, data compression can be the most beneficial. The increase in memory subsystem bandwidth from using hardware data compression was confirmed in the experiments showing the relative IPC increase in SPEC CPU2017 benchmarks up to 14 percent.

Keywords: processor architecture, memory subsystem, hardware data compression, network-on-chip, interprocessor links, processor model.

References

1. Serpa M.S., Moreira F.B., Navaux P.O., Cruz E.H., Diener M., Griebler D., Fernandes L.G. Memory performance and bottlenecks in multicore and GPU architectures. 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). IEEE, 2019. pp. 233–236.
2. Mohamed A.M., Mubark N., Zagloul S. Performance aware shared memory hierarchy model for multicore processors. Scientific Reports. 2023. vol. 13(1). no. 7313.
3. Iyer R., De V., Illikkal, R., Koufaty, D., Chitlur, B., Herdrich, A., Khellah M., Hamzaoglu F., Karl E. Advances in microprocessor cache architectures over the last 25 years. IEEE Micro. 2021. T. 41. № 6. C. 78–88.
4. Papazian I.E. New 3rd Gen Intel® Xeon® Scalable Processor (Codename: Ice Lake-SP). Hot Chips Symposium. 2020. C. 1–22.
5. Zhan J., Poremba M., Xu Y., Xie Y. NoΔ: Leveraging delta compression for end-to-end memory access in NoC based multicores. 19th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2014. pp. 586–591.
6. Deb D., Rohith M.K., Jose J. Flitzip: Effective packet compression for noc in multiprocessor system-on-chip. IEEE Transactions on Parallel and Distributed Systems. 2021. T. 33. № 1. pp. 117–128.

7. Wang Y., Han Y., Zhou J., Li H., Li X. DISCO: A low overhead in-network data compressor for energy-efficient chip multi-processors. Proceedings of the 53rd Annual Design Automation Conference. 2016. C. 1–6.
8. Wang Y., Li H., Han Y., Li X. A low overhead in-network data compressor for the memory hierarchy of chip multiprocessors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2017. vol. 37. no. 6. pp. 1265–1277.
9. Li X., Sondhi T. FlitReduce: Improving Memory Fabric Performance via End-to-End Network Packet Compression. UC Berkeley CS262A Report. 2021. 9 p.
10. Pullaiah T., Manjunathachari K., Malleswari B.L. BΔ-NIS: Performance analysis of an efficient data compression technique for on-chip communication network. Integration. 2023. vol. 89. pp. 83–93.
11. Pekhimenko G., Seshadri V., Mutlu O., Gibbons P.B., Kozuch M.A., Mowry T.C. Base-delta-immediate compression: Practical data compression for on-chip caches. Proceedings of the 21st international conference on Parallel architectures and compilation techniques. 2012. C. 377–388.
12. Gaur J., Alameldeen A.R., Subramoney S. Base-victim compression: An opportunistic cache compression architecture. ACM SIGARCH Computer Architecture News. 2016. vol. 44. no. 3. pp. 317–328.
13. Carvalho D.R., Seznec A. Understanding cache compression. ACM Transactions on Architecture and Code Optimization (TACO). 2021. vol. 18. no. 3. pp. 1–27.
14. Pekhimenko G., Seshadri V., Kim Y., Xin H., Mutlu O., Gibbons P.B., Kozuch M.A., Mowry T.C. Linearly compressed pages: A low-complexity, low-latency main memory compression framework. Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture. 2013. C. 172–184.
15. Young V., Kariyappa S., Qureshi M.K. CRAM: Efficient Hardware-Based Memory Compression for Bandwidth Enhancement. arXiv preprint arXiv:1807.07685. 2018.
16. Choukse E., Erez M., Alameldeen A.R. Compresso: Pragmatic main memory compression. 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2018. C. 546–558.
17. Surchenko A.V. [Evaluation of hardware data compression in interprocessor links of Elbrus processors]. Trudy Instituta sistemnogo programmirovaniya RAN – Proceedings of the Institute for System Programming of the RAS. 2022. vol. 34. no. 1. pp. 49–58.
18. Thuresson M., Spracklen L., Stenstrom P. Memory-link compression schemes: A value locality perspective. IEEE Transactions on Computers. 2008. vol. 57. no. 7. pp. 916–927.
19. Kozhin A.S., Surchenko A.V. Design of Data Compression Mechanism in Cache Memory of Elbrus Processors. International Conference Engineering and Telecommunication (En&T). IEEE, 2020. C. 1–5.
20. Nedbailo Y.A., Surchenko A.V., Bychkov I.N. Reducing miss rate in a non-inclusive cache with inclusive directory of a chip multiprocessor. Computer Research and Modeling. 2023. vol. 15. no. 3. pp. 639–656.
21. Nedbailo Y. Fast and scalable simulation framework for large in-order chip multiprocessors. 26th Conference of Open Innovations Association (FRUCT). IEEE, 2020. pp. 335–345.

Surchenko Alexander — Senior engineer, JSC "MCST"; Lecturer of the department, Department of computer science and computer engineering, MIPT. Research interests: performance and coherence support of memory subsystem of general-purpose processors, hardware data compression, cache memory. The number of publications — 10. Alexander.V.Surchenko@mcst.ru; 24, Vavilova St., 119049, Moscow, Russia; office phone: +7(499)135-8969.

Nedbailo Yuri — Ph.D., Lead engineer, JSC "MCST"; Lead engineer, JSC "INEUM named after I.S. Bruk". Research interests: memory subsystem of general-purpose processors. The number of publications — 25. yuri.nedbailo@mail.ru; 24, Vavilova St., 119049, Moscow, Russia; office phone: +7(916)936-8670.