УДК 004.72

ОТЕЧЕСТВЕННАЯ КОММУНИКАЦИОННАЯ СЕТЬ 3D-ТОР С ПОДДЕРЖКОЙ ГЛОБАЛЬНО АДРЕСУЕМОЙ ПАМЯТИ

А.А. Корж, Д.В. Макагон, А.А. Бородин, И.А. Жабин, Е.Р. Куштанов, Е.Л. Сыромятников, Е.В. Черемушкина

RUSSIAN 3D-TORUS INTERCONNECT WITH GLOBALLY ADDRESSABLE MEMORY SUPPORT

A.A. Korzh, D.V. Makagon, A.A. Borodin, I.A. Zhabin, E.R. Kushtanov, E.L. Syromyatnikov, E.V. Cheryomushkina

В статье рассматриваются детали реализации и первые результаты макетирования разработанной в НИЦЭВТ межузловой коммуникационной сети с топологией 3D-тор. Данная сеть может эффективно применяться как в вычислительных кластерах небольшого и среднего размера, так и в суперкомпьютерах транспетафлопсного уровня производительности. Особое внимание в статье уделено библиотеке параллельного программирования SHMEM, посредством которой программисту предоставляется доступ к глобально адресуемой памяти.

Ключевые слова: коммуникационная сеть, суперкомпьютер, 3D-тор, глобально адресуемая память

This paper gives the overview and early results of prototyping of the 3d-torus interconnect developed in NICEVT, Moscow. This interconnect was designed to be equally effective in small-size computing clusters and petascale systems. The key features of the interconnect are high fault-tolerance, high message rate per core supported by host adapter and hardware support for globally addressable memory provided via SHMEM parallel programming library.

Keywords: interconnection network, supercomputer, 3D-torus, globally addressable memory

Введение

Межузловая коммуникационная сеть является одной из важнейших частей суперкомпьютера, определяющей его производительность и масштабируемость. Разрыв тактовых частот процессора и подсистемы локальной и удалённой памяти постоянно увеличивается, поэтому именно время выполнения обращений в память удалённого узла становится «узким местом» при распараллеливании большинства задач на кластерах и суперкомпьютерах.

Коммуникационные сети современных суперкомпьютеров можно разделить на два класса: коммерческие (Ethernet, Infiniband, Quadrics, Myrinet) и заказные (IBM BlueGene, Cray XT, SGI Altix UV).

Коммерческие сети разрабатываются для широкого спектра применений, использование их в качестве коммуникационной сети суперкомпьютеров — лишь одно из них. В связи с этим в жертву универсальности приносится производительность в ряде режимов, в частности,

в коммерческих сетях не оптимизирована передача коротких сообщений, нет адаптивной передачи, аппаратно не поддержана отказоустойчивость при отказах линков.

Для систем среднего уровня (до 1000 узлов), а также для вычислительных кластеров, используемых в режиме счета нескольких небольших задач, умещающихся по потреблению памяти в одной стойке, чаще всего именно фактор цены оказывается важнее возможности эффективно считать одну задачу на всем суперкомпьютере. Как правило, для таких систем применяются коммерчески доступные коммуникационные сети, среди которых в последнее время с огромным отрывом лидируют сети стандарта Infiniband.

Для суперкомпьютеров с большим числом узлов (10–100 тысяч), достигших в настоящее время петафлопсного уровня производительности, вопрос сбалансированности производительности сети и процессоров является критическим, поэтому применение коммерческих технологий, таких как Infiniband, оказывается неприемлемым.

В связи с этим крупнейшие производители суперкомпьютеров, такие как Cray, IBM и SGI, используют коммуникационные сети собственной разработки: Cray Seastar2+ и будущая сеть машин серии Cray Baker — Gemini High-Speed Network, IBM Bluegene Torus Network, SGI NUMAlink. Такие сети недоступны для коммерческой продажи и используются исключительно в машинах высшего эшелона производительности.

В таких системах в настоящее время используются преимущественно сети с топологией 3D-тор [4]. Основные причины этого — трёхмерность коммуникационного шаблона многих физических задач, простая расширяемость системы, регулярность упаковки сети по стойкам с учетом экономии на межстоечных кабелях, хорошая масштабируемость и т.д.

При всех достоинствах топологии 3D-тор, она становится малоэффективной, когда число вычислительных узлов измеряется многими десятками тысяч (не говоря уже о сотнях тысяч). Переход к торам большей размерности хотя и увеличивает производительность (вместе со стоимостью), однако требует заметно более сложного способа упаковки и плохо соотносится с коммуникационными шаблонами физических задач.

В связи с этим в последнее время широко исследуются альтернативные топологии, в частности, различные варианты модифицированных деревьев (fat-tree) и сетей Клоса. Так, в числе последних разработок фирмы Cray — коммутатор YARC (для суперкомпьютера Cray X2), в ближайшем будущем планируется выпуск коммутатора Cray Aries (предположительно, с топологией dragonfly [5]).

Важным аспектом, определяющим применение заказных сетей, является поддержка глобально адресуемой памяти как основного режима программирования стратегических суперкомпьютеров. Данная парадигма может эффективно применяться при использовании обычных коммерческих микропроцессоров [1].

На отечественном рынке на данный момент присутствуют только зарубежные коммерческие разработки, не подходящие для суперкомпьютеров высшего эшелона производительности. Более того, каналы импорта компонентов зарубежных коммерческих сетей сравнительно легко могут быть перекрыты, чего с процессорами или памятью сделать практически невозможно. В связи с этим в последние несколько лет НИЦЭВТ ведет разработку заказной высокоскоростной коммуникационной сети в рамках проекта разработки суперкомпьютера стратегического назначения (СКСН).

Разрабатываемой коммуникационной сети уделяется особое внимание, как одному из критических компонентов СКСН. В настоящее время изготовлен, налажен и находится в опытной эксплуатации шестиузловой вычислительный кластер, построенный на основе второго поколения макетных образцов маршрутизаторов высокоскоростной коммуникационной сети с топологией 2D-тор на основе ПЛИС. Его тестирование специалистами ряда организаций показало хорошие результаты, сопоставимые с образцами коммерческой сети Infiniband DDR. При этом в наиболее тяжёлом режиме работы с интенсивным обменом короткими па-

кетами сообщений достигнутые характеристики существенно превосходили сеть Infiniband, в том числе и последнего поколения Infiniband QDR 4x.

В настоящее время изготовлен и находится в стадии отладки макетный образец третьего поколения, построенный на основе наиболее современных ПЛИС фирмы Xilinx; его характеристики в 1,8 раза лучше, достигаемый темп передачи сообщений — 14,9 миллионов пакетов в секунду.

В маршрутизаторе на аппаратном уровне поддерживаются четыре операции с удалённой памятью: асинхронное чтение (get), асинхронная запись (put), атомарное сложение (add), атомарное исключающее или (xor), а также различные методы синхронизации и контроль возврата выданных запросов на асинхронное чтение.

Реализована стандартная библиотека Cray SHMEM с дополнениями, позволяющими более эффективно выполнять типовые операции, и библиотека MPI на основе MPICH.

Далее в статье приводится описание разработанного макетного образца маршрутизатора и методология написания параллельных программ на языках C/C++ и Fortran с использованием аппаратно поддерживаемой в макете глобально адресуемой памяти. В заключении приведены основные полученные на текущий момент результаты и сформулированы планы на будущее.

1. Архитектура маршрутизатора

Разработка высокоскоростной коммуникационной сети в НИЦЭВТ ведётся уже несколько лет в рамках проекта создания СКСН. В 2007-м году был изготовлен полнофункциональный макет M2 с топологией 2D-тор, состоявший из шести узлов. Пропускная способность межузловых линков — 6,25 Гбит/с в каждую сторону, интерфейс с процессором — PCI-Express x4 (10 Гбит/с).

Обобщённая структурная схема разработанного маршрутизатора приведена на рис. 1. В его состав входят 4 линка (Link) — по два на каждое измерение тора, кроссбар (Crossbar), интерфейс с процессором вычислительного узла (NI+PCI) и сервисный процессор (Service PowerPC).

Каждый линк содержит в ПЛИС блок сериализатора-десериализатора и состоит из входной и выходной частей. Входная часть включает блоки виртуальных каналов и арбитры.

В маршрутизаторе реализован алгоритм бездедлоковой детерминированной маршрутизации, основанный на правилах «пузырька» и «порядка направлений» [3].

С помощью виртуальных каналов реализовано две независимых подсети для запросов и ответов, чтобы избежать дедлоков «процессор — сеть» (запросы не должны задерживать доставку ответов). Аппаратно поддерживается обход отказавших узлов (алгоритм «нестандартного первого и последнего шага») [2].

Выходная часть реализует надёжный протокол доставки сообщений, обеспечивающий непрерывную передачу пакетов и гарантирующий повтор в случае ошибок на линии.

Управление потоком данных осуществляется с помощью передачи кредитной информации (кредитов). Пакет передаётся следующему узлу, только если в принимающем буфере соответствующего виртуального канала гарантировано наличие необходимого свободного места. Такая гарантия предоставляется с помощью отсылки «кредитов»: как только в некотором входном буфере маршрутизатора освобождается место (один пакет передаётся в кроссбар и далее — соседнему узлу), по линку, связывающему данный входной буфер с выходным буфером другого маршрутизатора, отсылается «кредит» — информация об освободившемся месте. Основываясь на этой информации, в каждом маршрутизаторе определяется возможность передачи пакетов по линкам в каждый момент времени. «Кредиты»

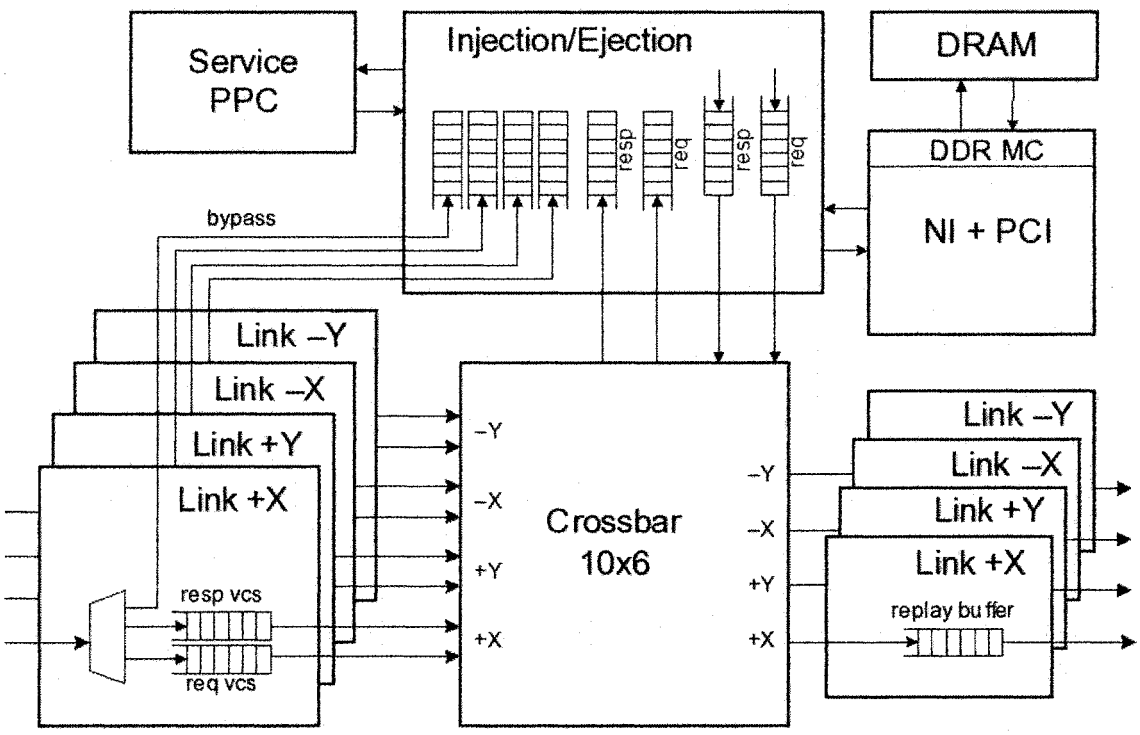

Рис. 1. Обобщённая структурная схема разработанного в НИЦЭВТ маршрутизатора

гарантируют наличие определённого количества свободного места (флитов), при этом за время передачи кредита свободное место может увеличиться (но гарантированно не уменьшится).

Предотвращение сбоев при передаче пакетов по линку обеспечивается собственным протоколом гарантированной передачи, реализованным поверх Xilinx Aurora link-layer protocol [6]. В случае несовпадения контрольной суммы при передаче пакета от одного маршрутизатора другому, пакет запрашивается повторно. В случае долговременного выхода из строя ряда линков возможна перенастройка таблиц маршрутизации сети и последующая перемаршрутизация пакетов. В сетевом интерфейсе содержится настраиваемая сервисной подсистемой таблица маршрутизации, позволяющая задавать различные маршруты между узлами.

Интерфейс с процессором вычислительного узла включает IP-ядро стандартного интерфейса PCI Express или HyperTransport, в зависимости от варианта исполнения маршрутизатора. Основной функцией интерфейса является инжекция пакетов в сеть. Для каждого пакета в автоматическом режиме на основании данных из встроенной таблицы маршрутизации определяется физический адрес и направление передачи. При приёме пакетов из сети интерфейс с процессором извлекает из полученных пакетов данные и записывает их напрямую в оперативную память вычислительного узла с использованием технологии прямого доступа в память (DMA). В некоторых случаях для повышения эффективности пакеты сначала накапливаются в памяти маршрутизатора, а уже затем записываются в память узла.

Для реализации подсистемы отказоустойчивости, позволяющей корректно обрабатывать отказы отдельных линков и даже вычислительных узлов, использовано встроенное в ПЛИС процессорное ядро, выполняющее функции коммуникационного процессора, следящего за состоянием сети и, при отказах, осуществляющего изменение данных в таблице маршрутизации.

Макетный образец третьего поколения, изготовленный в 2009 году, имеет пропускную способность линка 13 Гбит/с, интерфейс с процессором PCI Express x8. Поддерживается топология трёхмерный тор (используются 6 линков), адаптивная маршрутизация и агрегация коротких сообщений от разных узлов. Задержка между соседними узлами не превышает 1,2 мкс.

На аппаратном уровне поддерживаются четыре операции с удалённой памятью: асинхронное чтение (get), асинхронная запись (put), атомарное сложение (add), атомарное исключающее ИЛИ (xor), а также аппаратный контроль возврата выданных запросов на асинхронное чтение.

Для эффективной реализации операций с удалённой памятью потребовалось снизить накладные расходы, вносимые трансляцией адресов для поддержки виртуальной памяти. Было решено статически выделить область физической памяти, с которой может работать как маршрутизатор, так и пользовательская задача. Это позволило упростить интерфейс маршрутизатора, убрав громоздкие и неэффективные схемы трансляции адресов, присутствующие в коммерческих коммуникационных сетях, таких как Infiniband.

Одной из главных задач, поставленных перед разработчиками маршрутизатора для СКСН, было получение высокой пропускной способности на коротких пакетах. Для этого были использованы несколько виртуальных каналов; при передаче через PCI Express был применён режим Write Combining при посылке пакетов. При приёме пакетов была добавлена возможность агрегации коротких сообщений во встроенной памяти на плате маршрутизатора (через шину PCI данные сбрасываются большими блоками). На задачах типа умножения разреженной матрицы на вектор (SpMV) данные решения оказались довольно эффективными.

2. Программное обеспечение

Коммуникационная сеть, разработанная в НИЦЭВТ, имеет развитую программную поддержку, включающую реализацию стандартной библиотеки SHMEM (версии фирмы Cray с дополнениями, позволяющими более эффективно выполнять типовые операции) для языков C/C++ и Fortran. Также реализована версия библиотеки MPI 1.1 (MPICH), отлажена базовая реализация MPI-2, проведено функциональное тестирование с помощью пакета Intel MPI Benchmarks 3.2 (с проверкой на корректность доставки сообщений).

Поддерживается написание параллельных программ, использующих одновременно и MPI, и OpenMP, и SHMEM. Рекомендуется гибридный режим программирования SHMEM+OpenMP и SHMEM+OpenMP+MPI.

2.1. Библиотека SHMEM

Модели параллельного программирования можно разделить на два класса, в зависимости от того, на коммуникациях какого типа они основаны: двусторонних (передача сообщений) либо односторонних (обращения к удалённой памяти).

В двусторонних коммуникациях активное участие принимают две стороны: одна отправляет записываемое слово, а вторая — ждёт прихода слова, после чего копирует полученное слово из буфера приёма в нужную ячейку памяти. Адрес, куда записать слово в памяти второго узла, указывается самим получателем.

В односторонних коммуникациях активное участие принимает лишь инициатор: при записи он отсылает записываемое слово, а то, достигнув по сети адресата, напрямую записывается в память второго узла (при этом процессорное время на ожидание и запись вторым узлом не тратится). Адрес, куда записать слово в памяти второго узла, указывается отправителем.

Программы на MPI используют двусторонние коммуникации Send/Recv; переход к односторонним коммуникациям Put/Get (поддерживаемым не только интерфейсом SHMEM, но и стандартом MPI-2) потенциально способен повысить продуктивность разработки и сопровождения параллельных программ и эффективность их выполнения.

Такому переходу мешает в основном тот факт, что большая часть имеющихся библиотек написаны на MPI, большая часть разработанных параллельных алгоритмов созданы для двухсторонней парадигмы Send/Recv, большая часть программистов и математиков, занимающиеся распараллеливанием научных задач, привыкли думать именно в парадигме Send/Recv и не хотят переходить на другие парадигмы, не видя существенных преимуществ.

Система программирования SHMEM была разработана фирмой Cray более 15 лет назад, как интерфейс односторонних коммуникаций, способный стать эффективной альтернативой и дополнением к MPI и PVM. Интерфейс SHMEM поддерживается всеми MPP-системами фирмы Cray (Cray T3E, Cray XT5), Silicon Graphics (SGI Altix), интерконнектами Quadrics (QsNetIII). Также библиотека SHMEM (с некоторыми дополнениями) была реализована в системе МВС-Экспресс (под руководством А.О. Лациса).

По сути SHMEM реализует простейший вариант программирования в стиле PGAS (partitioned global address space). У каждого узла есть локальная память; каждому узлу также доступна удалённая память: узел может напрямую обращаться к локальной памяти любого узла системы. Поскольку обращения к удалённой памяти происходят через коммуникационную сеть, время их выполнения заметно больше, а темп — меньше, чем у обращений к локальной памяти. Ожидать выполнения каждой одиночной операции крайне дорого, поэтому требуется, чтобы программист явно выделял удалённые обращения.

В отличие от других PGAS-языков (например, UPC) SHMEM заставляет программиста явно выделять внешние обращения с помощью функций `shmem_put`, `shmem_get`, при этом дальнейшая группировка обращений и оптимизация выполняются аппаратно.

Большинство прикладных задач, использующих MPI, основано на простом коммуникационном шаблоне: preposted `MPI_Recv` + `MPI_Send`; однако такой шаблон гораздо естественнее записывается с использованием SHMEM как асинхронная запись (`shmem_put`) с подтверждением прихода сообщений. Подобным образом авторами статьи были переписаны на SHMEM тесты NPB CG и NPB UA.

Основу SHMEM составляют две операции: `shmem_put` — запись в память удалённого узла и `shmem_get` — чтение из памяти удалённого узла. Синхронизация происходит с помощью встроенной функции `shmem_barrier`. Возможность напрямую обращаться к удалённой памяти даёт большинство преимуществ работы с «общей памятью», не накладывая никаких дополнительных ограничений на то, как память распределена физически.

Главная проблема односторонних операций — необходимость при отправке указывать адрес в памяти другого узла. Пересылать адреса между узлами — неэффективно, поэтому было предложено следующее решение: на всех узлах использовать симметричные массивы (симметричные указатели), имеющие одинаковые адреса на всех узлах. Если в программе объявлен симметричный массив, то у каждого процесса будут свои локальные данные, однако адреса элементов и размер массива у всех процессов будут одинаковые. Благодаря этому процессы могут обращаться к массивам других узлов, используя локально известные адреса. Симметричное выделение памяти происходит с помощью функции `shmem_alloc`.

Помимо операций записи (`shmem_put`) и чтения (`shmem_get`), реализованы атомарные операции сложения (`shmem_add`) и исключающее ИЛИ (`shmem_xor`).

В макетах М2/М3 реализован SHMEM с поддержкой эффективной передачи коротких сообщений, атомарных операций и быстрого барьера, поэтому большинство алгоритмов (CG, FFT, UA) на SHMEM не только проще записываются, но и эффективнее выполняются.

2.2. Порядок сообщений и синхронизация узлов

При односторонних коммуникациях особое значение приобретают функции синхронизации и порядок сообщений.

В макете M2 используется модель программирования, основанная на односторонних коммуникациях без подтверждений; все пакеты при этом передаются детерминировано.

Данная модель предполагает стиль программирования, в котором каждая итерация алгоритма делится на две фазы: подкачка и счёт. Например: асинхронная подкачка данных для (i+1)-ой итерации; счёт i-ой итерации; ожидание завершения подкачки для (i+1)-ой итерации; асинхронная подкачка данных для (i+2)-ой итерации; счёт (i+1)-ой итерации и т.д.

Обычно подкачка выполняется с помощью операций чтения, однако стандартным приёмом программирования в стиле PGAS является замена подкачки с помощью чтений на посылку узлом-владельцем элементов с помощью операций записи. На множестве реальных задач (CG, UA) этот приём даёт значительный выигрыш, так как снижает нагрузку на сеть, потому как операция чтения посылает по сети два пакета (запрос и ответ), а операция записи на многих системах реализована посылкой пакета только в одну сторону.

Ожидание завершения подкачки (с помощью операций `shmem_put`, `shmem_get`, `shmem_add` и `shmem_xor`) может осуществляться стандартным методом — с помощью вызова общего барьера (`shmem_barrier`). При этом также реализованы несколько специальных методов, функционально заменяющих барьер для контроля выполнения выданных асинхронных записей и чтений, позволяющих достичь намного большей производительности.

Контроль возврата выданных запросов на асинхронное чтение реализуется блокирующей функцией `shmem_syncreads` (используется аппаратно реализованный счётчик в сетевом интерфейсе: при отправке запроса на чтение он увеличивается на единицу, при получении ответа — уменьшается на единицу).

Контроль порядка выданных асинхронных записей реализуется функцией `shmem_fence`: гарантируется, что все отосланные до `shmem_fence` сообщения узлом A узлу B достигнут адресата строго до сообщений, отосланных после `shmem_fence`. Таким образом, порядок гарантируется только для пакетов, имеющих одинаковых отправителей и получателей. При детерминированной маршрутизации такой порядок соблюдается по определению и не требует какой-либо синхронизации узлов.

Контроль выполнения выданных асинхронных записей осуществляется с помощью блокирующей функцией `shmem_quiet`. Стандартный метод реализации — потребовать подтверждений для всех операций. Если пакеты были отосланы детерминировано, то достаточно запросить подтверждения только последней отправленной операции для каждого узла. Часто именно с помощью этой функции гарантируется глобальный порядок сообщений: ни одно сообщение, отосланное до `shmem_quiet`, не должно обогнать сообщение, отосланное после `shmem_quiet`.

Функция `shmem_barrier` гарантирует, что каждый процесс продолжит работу только после того, как все узлы дойдут до места вызова барьера в коде программы. Данная функция может быть реализована в две фазы: `shmem_barrier_notify` (начало барьера — неблокирующая) и `shmem_barrier_wait` (ожидание окончания барьера — блокирующая).

В большинстве реализаций SHMEM барьер выполняет важную упорядочивающую функцию: узлы выйдут из барьера только тогда, когда каждый узел получит все адресованные ему сообщения.

Однако данное условие является слишком строгим и во множестве (даже в большинстве) задач совсем не требуется. Вполне достаточно, чтобы узел выходил из барьера, как только он получит все отосланные ему до барьера сообщения. При таком условии каждый узел может

продолжить счёт сразу, как только получит все адресованные ему данные, не дожидаясь остальных узлов (вплоть до следующего барьера). Такой «быстрый» барьер в среднем, как минимум, в два раза быстрее; именно он реализован в функции `shmem_barrier` библиотеки SHMEM для макета M2/M3. Обычный «медленный» барьер реализуется выполнением двух «быстрых» барьеров подряд.

Помимо глобального барьера, в котором участвуют все процессы данной задачи, поддерживаются барьеры по заданной группе, где явно указывается (с помощью маски, шага) множество узлов, участвующих в барьере.

3. Результаты тестирования, производительность

3.1. Intel MPI Benchmarks

Intel MPI Benchmarks (IMB) 3.2 — свободно распространяемый набор тестов, предназначенный для оценки эффективности выполнения операций MPI (с проверкой на корректность доставки сообщений) [7]. Ранее данный набор тестов был известен как Pallas MPI Benchmarks (PMB).

На макете M2 на тесте PingPong на сообщениях размером до 2 килобайт пропускная способность находится на уровне Infiniband DDR 4x. На больших сообщениях из-за более медленного линка M2 отстает. Макет M3 превосходит Infiniband QDR 4x на коротких сообщениях и несколько уступает на больших сообщениях (рис. 2).

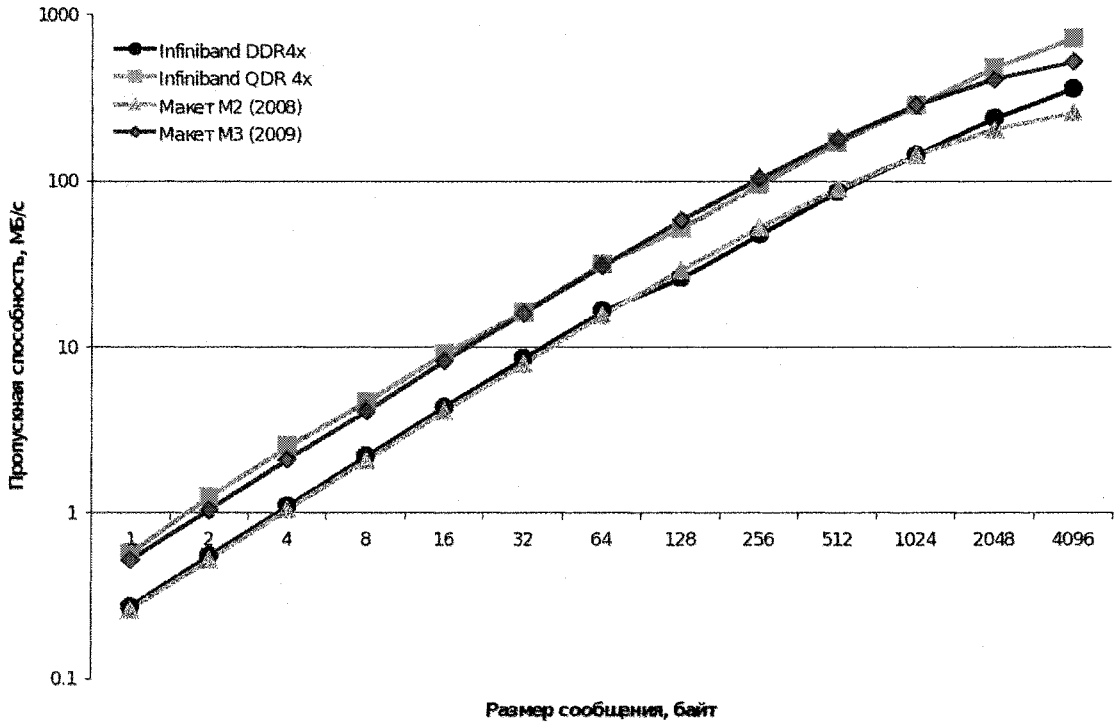

Рис. 2. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте IMB PingPong

На тесте Bcast коммуникационная задержка при сообщениях размером до 1 килобайт на макете M2 на 70% меньше, чем на Infiniband DDR 4x. На больших сообщениях задержка на M2 превосходит уровень Infiniband (рис. 3).

По характеристикам (на всех тестах из пакета IMB) на пакетах до 2КБ реализация MPI на M2 не отстает от реализации MPI на сети Infiniband DDR 4x. На больших пакетах производительность ниже по двум причинам: у M2 существенно меньшая пропускная способность сетевых линков, а также отсутствует поддержка передачи больших сообщений в режиме DMA.

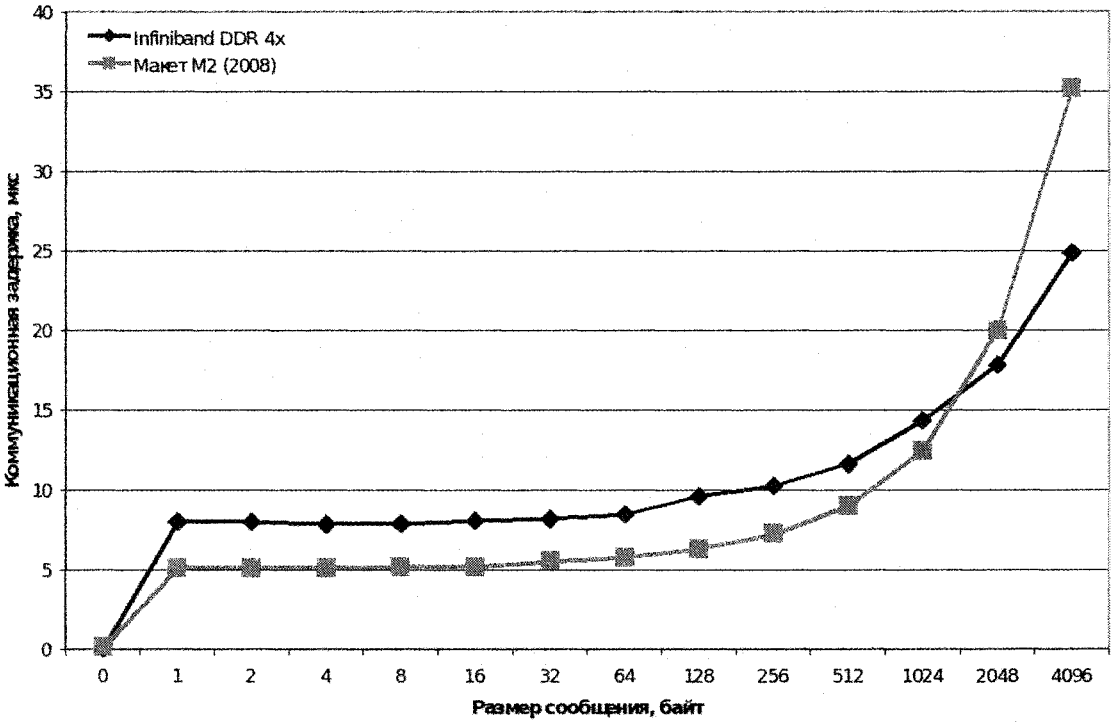

Рис. 3. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте IMB Bcast

3.2. HPCC RandomAccess

RandomAccess — один из семи тестов из набора HPC Challenge Benchmark [7]. Тест характеризует производительность вычислительной системы при наиболее неблагоприятном режиме доступа к памяти.

Тест имеет коммуникационный шаблон все-всем при минимальном размере пакета 8 байт; результатом теста является достигаемое число обновлений ячеек памяти по случайным адресам в секунду.

На макете M2 результаты более чем в 2 раза превышают результаты Infiniband QDR 4x в случае использования API нижнего уровня IB Verbs, и более чем в 10 раз в случае использования MPI (рис. 4).

3.3. Барьерная синхронизация

Барьерная синхронизация — наиболее важная синхронизационная функция, используемая в MPI и SHMEM.

На макетах M2 и M3 до 4-х узлов барьерная синхронизация работает на уровне Infiniband QDR 4x, на большем числе узлов Infiniband QDR 4x значительно проигрывает — выполняется в несколько раз медленнее (рис. 5).

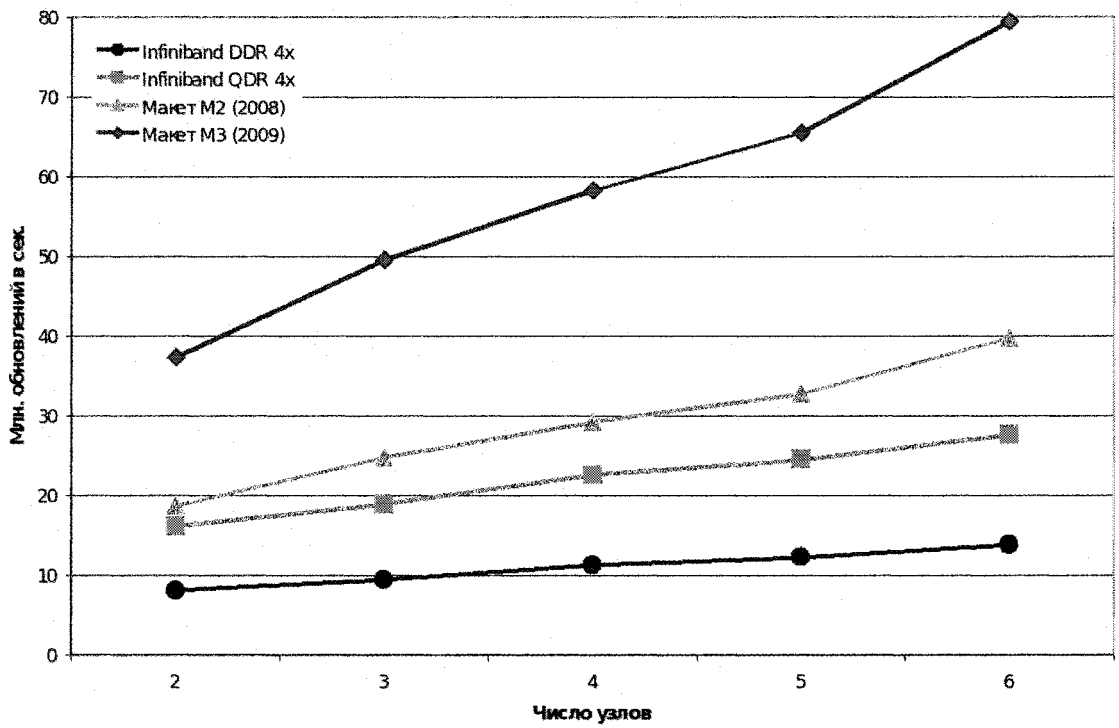

Рис. 4. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте HPCC RandomAccess

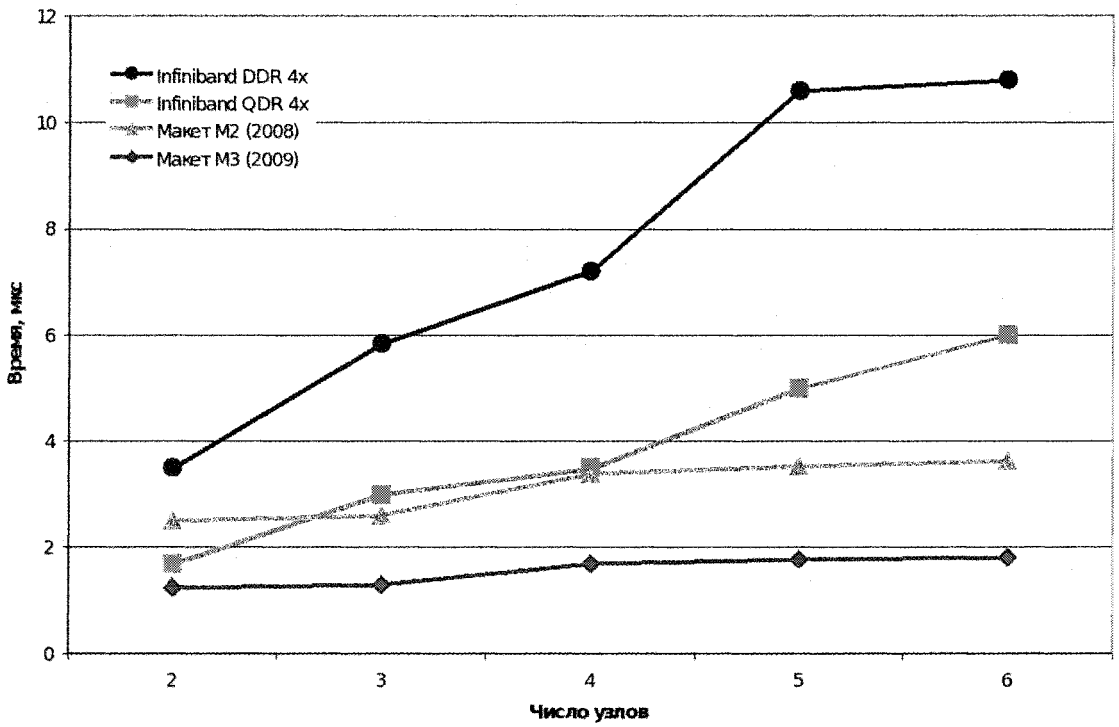

Рис. 5. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте времени выполнения барьерной синхронизации

3.4. NAS Parallel Benchmarks

NAS Parallel Benchmarks (NPB) — набор вычислительных задач, предназначенный для измерения производительности суперкомпьютеров в различных актуальных научных приложениях [8]. Набор включает тесты Conjugate Gradient (CG), Fast Fourier Transform (FT), MultiGrid (MG), Integer Sort (IS), Unstructured Adaptive (UA), Data Traffic (DT) и т.д.

Все тесты написаны на языке Fortran 77; существуют параллельные реализации NPB, использующие MPI, OpenMP, HPF, однако версий, использующих односторонние коммуникации в парадигме PGAS — нет.

Авторами статьи были реализованы с использованием SHMEM два теста — CG и UA. Ядром теста CG [8] является итеративное умножение разреженной матрицы на вектор. Поскольку реализованное в NPB блочно-циклическое разбиение матрицы специально нацелено на увеличение доли нерегулярных коммуникаций, на малом числе узлов производительность версии на SHMEM оказалась сравнима с версией на MPI. Оптимизированная версия, в которой используется блочное разбиения матрицы по строкам, позволяет более эффективно реализовать алгоритм на SHMEM, что дачт прирост в среднем на 36%.

В тесте UA [9] решается задача Дирихле уравнения теплопереноса в трехмерной кубической области на нерегулярной декартовой сетке. Источник тепла представляет собой шар, движущийся с постоянной скоростью. Для решения применяется нерегулярная сетка, причём каждые несколько шагов происходит её адаптация: в областях с большим градиентом температуры сетка измельчается, с малым — укрупняется. Таким образом, тест нацелен на измерение производительности при нерегулярном динамически изменяемом шаблоне доступа к памяти.

Тест UA, появившийся в NPB 3.1, до этого распараллеливался лишь с использованием OpenMP, распараллеливание в модели MPI для данной задачи никем успешно не произведено до сих пор (т.е. невозможно на данный момент получить результаты об ускорении на системах с Infiniband), поэтому полученные результаты для SHMEM представляют особый интерес.

Ускорение на макете M2 для версии на SHMEM на 6-ти узлах (ядрах) составляет 5,4 раза (рис. 6); ускорение OpenMP-версии на других системах для 6-ти ядер не превосходит 2,8 раз; для 16-ти ядер — 4,4 раз. Ускорение измерялось для класса C относительно последовательного времени выполнения теста, которое для макета M2 составляло 1527 с, для MVS-Express — 1292 с, для SGI Altix 3700 — 1585 с (OpenMP версия).

4. Заключение

Разработанная коммуникационная сеть 3D-тор с поддержкой глобально адресуемой памяти может быть использована как в кластерах, так и в суперкомпьютерах транспетафлопсного уровня производительности.

Коллективом программистов были адаптированы библиотеки MPI 1.1, SHMEM и MPI-2, а также тесты IMB, NetPIPE. Благодаря аппаратной поддержке односторонних коммуникаций авторам удалось эффективно реализовать тесты NPB CG и UA, используя модель программирования SHMEM.

Модель программирования SHMEM, будучи простейшим представителем семейства языков в парадигме PGAS, является более продуктивной и эффективной, чем модель MPI, при этом она не содержит серьёзных недостатков более сложных языков PGAS, таких как UPC и Co-array Fortran. Данная тема будет подробно рассмотрена в следующей статье. Модель программирования SHMEM обладает достаточной гибкостью и функциональностью, чтобы реализовать все остальные языки PGAS.

В настоящее время успешно используется макетный образец сети второго поколения

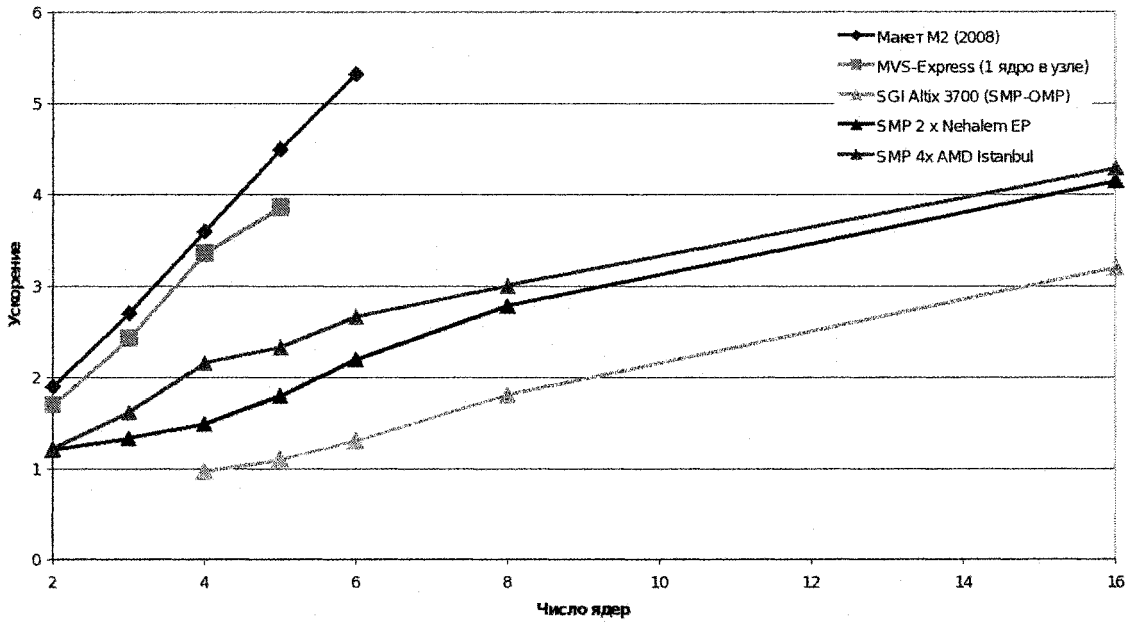

Рис. 6. Ускорение на тесте NPB UA от числа используемых ядер

(М2), заканчивается тестирование макета третьего поколения (М3) и уже ведутся работы по созданию макета четвёртого поколения (М4).

Макет третьего поколения включает в себя эффективную поддержку коллективных операций broadcast и allreduce (по кольцам и измерениям тора), кольцевые буфера в памяти маршрутизатора, поддержку «активных сообщений», RPC и подтверждений.

Макет четвёртого поколения сможет адекватно конкурировать с последними разработками как отечественных фирм, так и зарубежных компаний (в том числе с Infiniband EDR и Cray Gemini). Он будет включать отдельные сети для глобальной синхронизации и коллективных операций (с древовидной топологией), также будет произведена замена интерфейса PCI-Express на более производительный HyperTransport. Параллельно с этим ведутся работы по разработке блейдов под процессоры AMD Opteron, на которые будет интегрирован кристалл коммуникационной сети и мультитредовый ускоритель. Разрабатывается специализированная ОС суперкомпьютера стратегического назначения на базе ядра ОС Linux.

Коллектив разработчиков выражает признательность Л.К. Эйсымонту — за формирование первоначального внимания и интереса к тематике высокоскоростных коммуникационных сетей.

Статья рекомендована к печати программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010» http://agora.guru.ru/pavt.

Работа выполнена при поддержке РФФИ, грант №09-07-13596-офи_ц.

Литература

1. Evaluating NIC hardware requirements to achieve high message rate PGAS support on multicore processors / Keith D. Underwood, Michael J. Levenhagen, Ron Brightwell // Proceedings of the 2007 ACM/IEEE conference on Supercomputing. – 2007. – P. 31 – 40.

2. Scott, S. Synchronization and communication in the T3E multiprocessor / Steven L. Scott //

Proceedings of the seventh international conference on Architectural support for programming languages and operating systems. – 1996. – P. 26 – 36.

3. Duato, J. Interconnection Networks, An Engineering Approach / J. Duato, S. Yalamanchili, L. Ni. – IEEE Computer Society Press, 1997. – 812 p.
4. Джосан, О.В. Организация коммуникационной сети для транспетафлопсных суперкомпьютеров / О.В. Джосан, А.А. Корж // Труды ИСА РАН: динамика неоднородных систем. – 2008. – Т. 32, № 3. – С. 267 – 274.
5. Technology-Driven, Highly-Scalable Dragonfly Topology / J. Kim, W.J. Dally, S. Scott, D. Abts // Computer Architecture. – 2008. – № 08. – С. 77 – 88.
6. Aurora 8B/10B Protocol Specification SP002 (v2.1) [Электронный ресурс]. – Xilinx, Inc, 2009. URL: http://www.xilinx.com/products/design_resources/conn_central /grouping/aurora.htm.
7. Saini, S. Performance evaluation of supercomputers using HPCC and IMB Benchmarks / S. Saini et al. // Journal of Computer and System Sciences. – 2008. – № 74. – P. 965 – 982.
8. Jin, H. The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance / H. Jin, M. Frumkin, J. Yan // NAS Technical Report NAS-99-011 October 1999. – 72 p.
9. Unstructured adaptive (UA) NAS Parallel Benchmark / M. Field, H. Feng, R.F. Van der Wijngaart, R. Biswas // NASA Ames Research Center, CA. – 2004. – 17 p.

Макагон Дмитрий Викторович, научный сотрудник, отдел «Высокоскоростные коммуникационные сети», ОАО «Научно-исследовательский центр электронной вычислительной техники», makagond@mail.ru.

Корж Антон Александрович, начальник отдела, отдел «Высокоскоростные коммуникационные сети», ОАО «Научно-исследовательский центр электронной вычислительной техники».

Поступила в редакцию 7 апреля 2010 г.