И. А. Адамович, А. В. Климов, Ю. А. Климов, А. Ю. Орлов, А. Б. Шворин, Опыт разработки коммуникационной сети суперкомпьютера «СКИФ-Аврора», *Программные системы: теория и приложения*, 2010, том 1, выпуск 3, 107–123

ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № 3(3), 2010, с. 107–123

УДК 004.7

И. А. Адамович, А. В. Климов, Ю. А. Климов, А. Ю. Орлов, А. Б. Шворин

Опыт разработки коммуникационной сети суперкомпьютера «СКИФ-Аврора»

Аннотация. В данной статье обсуждается опыт реализации коммуникационной сети с топологией 3D-тор для суперкомпьютера «СКИФ-Аврора». Авторами выполнена разработка всех уровней сетевой инфраструктуры от схемотехники сетевых адаптеров, реализованных в ПЛИС, до пользовательских библиотек. Приводятся соображения о целесообразности самостоятельной разработки суперкомпьютерных сетей в сравнении с покупкой готовых решений. Показано, что по некоторым параметрам общедоступные на рынке сетевые решения далеки от совершенства, и некоторые их характеристики могут быть значительно превзойдены при самостоятельной разработке. В качестве иллюстрации приводится сравнение нашей разработки с сетью InfiniBand по темпу выдачи сообщений, и обсуждаются некоторые технические приемы, позволившие добиться высокого результата.

Ключевые слова и фразы: суперкомпьютеры, сеть, ПЛИС, маршрутизация, 3D-тор.

Введение

Коллектив авторов данной работы занимается разработкой и реализацией коммуникационной сети топологии 3D-тор для суперкомпьютера «СКИФ-Аврора» [1]. Под словами «разработка и реализация сети» здесь следует понимать изготовление всех уровней сетевой инфраструктуры: от схемотехники сетевых адаптеров до интеграции с операционной системой и поддержки коммуникационных библиотек, таких как MPI [17] и SHMEM [18].

В данной статье мы хотели бы обсудить целесообразность самостоятельной разработки сетей для суперкомпьютеров и выделить преимущества, которые можно при этом получить по сравнению с покупкой готовых решений, таких как InfiniBand. Опираясь на наш

Работа выполняются по научно-технической программе Союзного государства «СКИФ-ГРИД», а также при поддержке РФФИ по проекту № 09–07–13598–офи_ц.

опыт, мы покажем, что, используя современные микросхемы программируемой логики (ПЛИС), разработчик суперкомпьютера имеет возможность с относительно небольшими трудозатратами создавать сетевые решения, в целом не уступающие доступным на рынке. При этом ПЛИС позволяют легко специализировать решение под задачу и добиться значительного преимущества по наиболее важным для задачи характеристикам сети.

Мы продемонстрируем, что предложенное нами решение существенно превосходит InfiniBand по такому параметру как темп выдачи сообщений, и поясним, почему этот параметр играет важную роль при решении на суперкомпьютере достаточно широкого класса задач.

1. О целесообразности самостоятельной разработки сетей

С определенной точки зрения, сеть является самым важным компонентом суперкомпьютера. В самом деле, если желающего запустить задачу интересует лишь совокупная вычислительная мощность процессоров, то он вполне может использовать такие технологии, как BOINC [16] (на которой реализован известный проект SETI@HOME), MapReduce, облачные вычисления и т. п. Общие черты этих технологий — слабая связность при высокой вычислительной мощности и низкой по сравнению с суперкомпьютерами цене. Очевидно однако, что такой подход имеет лишь ограниченное применение, и в некоторый момент именно способность к обмену данными между вычислительными узлами становится узким горлом, препятствующим эффективному выполнению задач.

Именно поэтому производители суперкомпьютеров по всему миру уделяют коммуникациям пристальное внимание. В частности, имеет место тенденция к сверхплотной упаковке вычислительных ядер и, вообще, уход от традиционной кластерной архитектуры. Выигрыш при таком подходе достигается за счет повышения коммуникационной связности вычислителей, несмотря на связанные с уплотнением сложности отвода тепла. На пике вычислительных возможностей всегда были и остаются машины с заметно развитыми по сравнению с конкурентами сетями — машины фирм Cray, SGI и IBM.

В связи с тем, что коммуникационные сети являются стратегическим компонентом, у отечественного производителя суперкомпьютеров возникают определенные трудности. Прежде всего, сети, которые входят в состав машин, находящихся на вершине мирового рейтинга,

просто-напросто не продаются. Ни одна из перечисленных фирм — лидеров суперкомпьютерного сообщества не продаст свою сеть отдельно от машины. И даже целиком машину, занимающую одну из верхних строчек в мировом рейтинге, купить в России тоже нельзя. Далее, об устройстве сетей, как рекордных, так и доступных на рынке, известно очень мало. Это означает, что у производителя суперкомпьютеров есть всего два варианта: либо покупать общедоступную сеть как, в некотором роде, «кота в мешке», надеясь, что она хорошо себя покажет в его машине и именно на его задачах, либо попытаться реализовать свою сеть.

К счастью, последнее стало вполне доступно по трудозатратам практически любому, кто берется собирать суперкомпьютер, и не требует в обязательном порядке изготовления собственных микросхем. Кроме нашего решения, использующего технологию ПЛИС, о котором будет рассказано ниже, можно привести в качестве примера машину «МВС-Экспресс», созданную в ИПМ РАН совместно с НИИ «Квант» [4], для которой разработана сеть на основе прямой коммутации PCI-Express при помощи стандартных микросхем фирмы PLX.

Мы утверждаем, что стоит попытаться сделать свою сеть хотя бы для того, чтобы понимать, как устроены современные коммуникационные сети, какие проблемы приходится решать их разработчикам, и на основании этого делать правильный выбор при покупке готового решения. Однако оказывается, что не так уж сложно создать сеть, в целом не уступающую доступным на рынке, а по некоторым характеристикам значительно их превосходящую. И это несмотря на то, что коммерчески доступные решения таких фирм как Melanox и QLogic постепенно сокращают свое отставание в производительности от решений перечисленных выше лидеров.

Итак, давайте рассмотрим, за какие же характеристики сети имеет смысл бороться разработчикам. Основными параметрами, определяющими производительность сети, являются *пропускная способность* (bandwidth) и *коммуникационная задержка* (latency). Значения этих параметров публикуются на сайтах разработчиков сетей как наиболее важные; кроме того, существует множество тестов, предназначенных для их измерения [11–13].

Пропускная способность определяется как количество информации, которое может быть передано от узла к узлу в единицу времени. Заметим, что при передаче больших объемов информации, пропускная способность практически полностью определяется физическими

характеристиками канала, и конкурировать здесь по существу невозможно: все производители сетей имеют возможность пользоваться одними и теми же трансиверами, кабелями и остальным оборудованием, доступным на сегодняшний день.

Задержка определяется как время от начала посылки сообщения до его получения на другом узле. От посылки до получения сообщению, в общем случае, приходится пройти через несколько «перевалочных пунктов»: от процессора-источника к сетевому адаптеру, сквозь сетевую инфраструктуру (маршрутизаторы, свитчи или что-то подобное), от сетевого адаптера на приемном конце к процессору, при этом в каждом из этих случаев возможны различные преобразования сообщения в соответствии с протоколами передачи данных, а также задержки на ожидание возможности отправки в следующий пункт. Здесь очевидны два направления для оптимизации:

(1) Адаптация топологии сети под задачу с целью уменьшить накладные расходы на прохождение сетевой инфраструктуры. Например, сеть нашей разработки имеет топологию 3D-тор, которая хорошо подходит для решения многих физических задач из нашего трехмерного мира, когда основной обмен данными происходит между соседними в смысле сетевой топологии узлами.

(2) Оптимизация протоколов, используемых для передачи данных на всех уровнях; возможно, специализация их под задачу, с целью сократить накладные расходы на преобразование формата сообщения, передачу заголовков и другой служебной информации и т. п.

Действительно, и пропускная способность, и задержка во многом определяют эффективность сети на реальных задачах. Однако есть класс задач, где требуется послать сразу очень много небольших сообщений, возможно разным адресатам. В этом случае на первый план выходит такая характеристика сети как *темп выдачи сообщений* (message rate), на которой мы остановимся подробнее.

2. Темп выдачи сообщений

Темп выдачи определяется как количество сообщений, которое узел может выдать в сеть в единицу времени. Вообще говоря, значение темпа выдачи r зависит от размера сообщения L:

$$r = r(L) = N/t,$$

где за измеренное время t было выдано N сообщений. При этом нужно проводить усреднение по достаточно длительному промежутку

времени, иначе мы измерим скорость заполнения входных буферов. В некоторых случаях удобней говорить о времени выдачи одного сообщения — величине обратной к темпу выдачи, $\tau(L) = 1/r(L)$.

2.1. Темп выдачи и эффективность сети

Легко видеть, что значение темпа выдачи принципиально ограничено пропускной способностью сети. Действительно, сообщение длины L не может быть передано быстрее, чем за время L/B (где B — пиковая пропускная способность канала), и, таким образом,

$$r(L) \leq B/L.$$

В реальности же большинству сетей на малых сообщениях не удается даже приблизиться к этому значению. Причина кроется в накладных расходах на посылку каждого сообщения, и это довольно серьезная проблема на пути к созданию эффективных сетей.

Как и в случае с коммуникационной задержкой, накладные расходы связаны с протоколами передачи информации на пути сообщения от одного вычислительного ядра к другому. Во-первых, при малом размере сообщений значительную часть передаваемых данных (возможно, даже заметно превосходящую по размеру непосредственно «полезные» данные) занимает различного рода метаинформация, такая как заголовки сетевых пакетов, контрольные суммы и т. д. Чем более развесист и универсален протокол, тем больше придется передавать метаинформации, и здесь у специализированных сетей есть большое преимущество перед сетями общего назначения, такими как InfiniBand. Во-вторых, универсальные протоколы часто предполагают ожидание ответа от приемника информации, тем самым заставляя отправителя простаивать и не давая ему выбрать всю имеющуюся пропускную способность канала. Реализация специализированных протоколов позволяет свести такие задержки к минимуму. Например, в сети нашей разработки, отправка короткого сообщения другому вычислительному ядру делается выполнением всего одной процессорной инструкции MOV. Как будет видно из графиков ниже, это позволяет нам полностью выбрать пропускную способность наших физических линков уже на самых коротких сообщениях.

Почему значение темпа выдачи важно с точки зрения эффективности приложения? При низком темпе выдачи ядро процессора будет вынуждено часто простаивать в ожидании возможности отправить готовое сообщение. Существует целый класс задач с нерегулярной

интенсивной коммуникационной нагрузкой, которые весьма чувствительны к темпу выдачи. Для оценки эффективности суперкомпьютера на этом классе задач часто используют ставший уже классическим тест RandomAccess (GUPS) из набора HPC Challenge Benchmark [12]. Если характеристики сети (пропускная способность, темп, задержка) ниже соответствующих характеристик памяти на узлах (а это верно для всех современных суперкомпьютеров), то показатели на тесте RandomAccess определяются в основном темпом выдачи. Таким образом, высокие показатели темпа выдачи коротких сообщений чрезвычайно важны для оценки качества машины.

Если сеть суперкомпьютера имеет низкий темп выдачи сообщений, можно пытаться это обойти. Понятно, что при посылке сообщений большого размера накладные расходы на посылку сообщения становятся менее заметными, и реальная пропускная способность приближается к пиковой. Поэтому, самый простой и наивный способ повысить темп выдачи — это завести огромный буфер на входе в сеть и накапливать в нем мелкие сообщения, агрегируя их в более крупные. Однако этот способ имеет ряд существенных недостатков.

- Прежде всего хочется отметить, что возможность посылки множества мелких сообщений делает удобным использование совершенно отличных от традиционного MPI стилей программирования. В первую очередь, это односторонние коммуникации, давно используемые фирмами Cray и SGI в виде библиотеки SHMEM [18], и теперь вошедшие в стандарт MPI-2 [17]. Дисциплина односторонних коммуникаций является более высокоуровневым средством по сравнению с дисциплиной Send-Receive и дает возможность решать принципиально более сложные задачи. В качестве примера можно привести работу А. А. Коржа [6], использовавшего библиотеку SHMEM для распараллеливания теста NPB UA [13] на тысячи ядер суперкомпьютера IBM Blue Gene/P. Этот тест представляет из себя вычисления на неструктурной адаптивной сетке, и до указанной работы имел лишь реализацию с использованием OpenMP для машин с общей памятью. В отличие от большинства других тестов бенчмарка NPB, этот тест не был реализован специалистами NASA в традиционной парадигме Send-Receive, надо полагать, в силу своей сложности.
- В парадигме односторонних коммуникаций понятие барьера имеет несколько другой смысл, чем традиционный барьер в MPI-1: барьер должен гарантировать, что все находящиеся в полете сообщения были доставлены до адресатов. Реализация такого барьера

```
MPI_Barrier()
for (i = 0; i < /*количество итераций*/ N; i++)
  if (my_id == 0)
    MPI_SEND(/*длина сообщения*/ L, 1 /*получатель*/);
  else if (my_id == 1)
    MPI_RECV(/*длина сообщения*/ L, 0 /*отправитель*/);
MPI_Barrier()
```

Рис. 1. Тест MsgRate

предполагает посылку коротких служебных сообщений всем от всех (All-to-All). И это как раз то место, где агрегация коротких сообщений в более крупные не помогает. На сетях с хорошим темпом выдачи барьер будет работать хорошо, на сетях с низким — плохо.

- С ростом количества вычислительных узлов в машине, возможность агрегировать сообщения для каждого отдельного узла падает.
- И, наконец, агрегация увеличивает коммуникационную задержку. Для эффективных параллельных вычислений требуются высокореактивные сети, обладающие как малой задержкой, так и высоким темпом выдачи. Достижение этой цели требует определенных усилий; даже такое успешное решение как InfiniBand демонстрирует довольно посредственные значения темпа выдачи на малых сообщениях.

2.2. Методика измерения темпа выдачи сообщений

Для измерения характеристик сети мы использовали тесты [14], входящие в поставку библиотеки ScaliMPI. Тест Ping-Pong измеряет задержку сети, а для измерения темпа выдачи сообщений применялся тест MsgRate[1], который представлен на рисунке 1.

Здесь один процесс (с номером 0) только отправляет сообщения, а другой (с номером 1) только принимает сообщения. Измеряется общее время выполнение теста. Полученное значение, деленное на количество итераций, — это время выполнения одной посылки, то есть величина, обратная темпу выдачи сообщений. Размер сообщения по отношению ко времени выполнения одной итерации — это реальная пропускная способность сети для сообщений данного размера.

[1]Этот тест входит в поставку ScaliMPI под именем «Ping-Ping», что может ввести в заблуждение, поскольку в наборе IMB [11] есть более известный тест с тем же именем Ping-Ping, который устроен совершенно иначе. Во избежание путаницы мы решили дать свое название тесту.

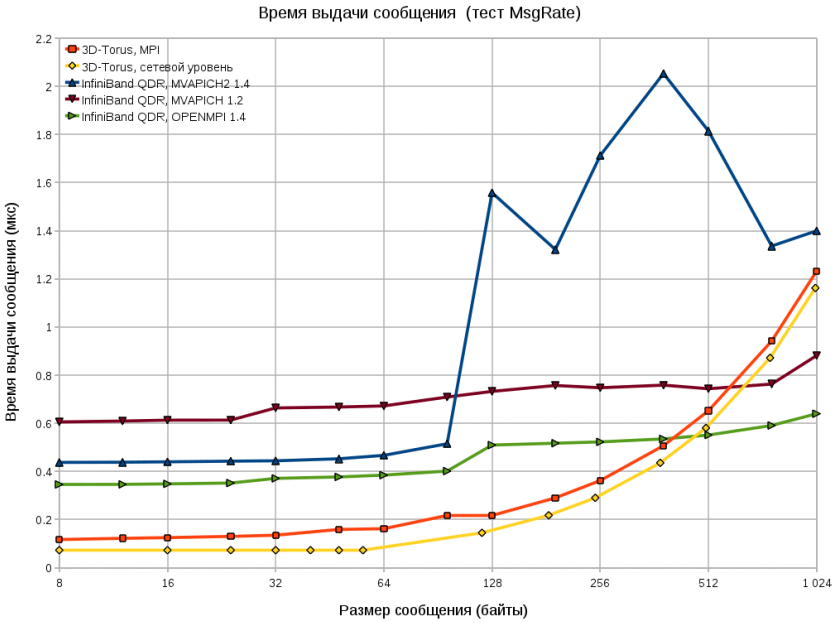

Рис. 2. Время выдачи сообщений (меньше — лучше)

2.3. Результаты измерений для сетей суперкомпьютера «СКИФ-Аврора»

На приведенных графиках представлены результаты измерений для коммуникационных сетей 3D-тор и InfiniBand QDR суперкомпьютера «СКИФ-Аврора». Рис. 2 показывает время выдачи одного сообщения (величину обратную к темпу выдачи), а рис. 3 — реальную пропускную способность на данном тесте. Результаты разнятся для различных реализаций MPI поверх InfiniBand, но видно, что во всех случаях темп выдачи сообщений размером до 256 байт весьма невысок по сравнению с сетью 3D-тор. С увеличением размера сообщений темп выдачи у 3D-тора падает по той причине, что суммарный объем выдаваемых в единицу времени сообщений достигает пиковой пропускной способности линка. Для сети InfiniBand аналогичное насыщение достигается лишь на сообщениях гораздо большего размера.

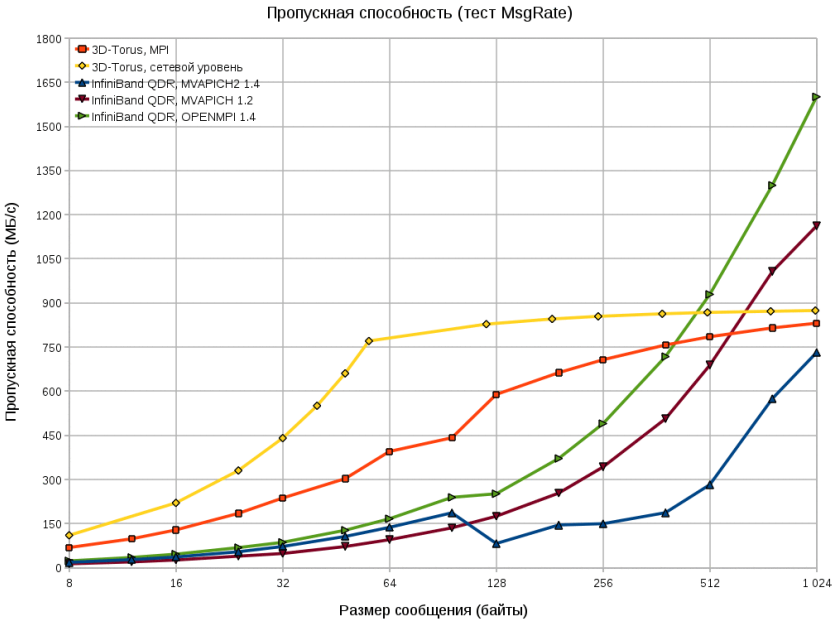

Рис. 3. Пропускная способность на тесте MsgRate (больше — лучше)

Что касается задержки (графики на рис. 4 и 5), то обе сети показывают близкие результаты на коротких сообщениях (8–64 байта). На сообщениях средней длины (128–2048 байтов) сеть 3D-тор заметно лучше. На длинных сообщения InfiniBand начинает обгонять 3D-тор за счет существенно большей пиковой пропускной способности.

Результаты тестирования приведены для соединения точка-точка двух узлов суперкомпьютера «СКИФ-Аврора». Для сообщений каждого размера тесты проводились в течение достаточного времени для насыщения всех имеющихся входных, выходных и внутрисетевых буферов. Технические характеристики узла таковы:

- Два 4-ядерных процессора Intel Xeon CPU X5570 2.93GHz.
- Пиковая производительность узла: 93.76 Гфлоп/с.
- Интерконнект:

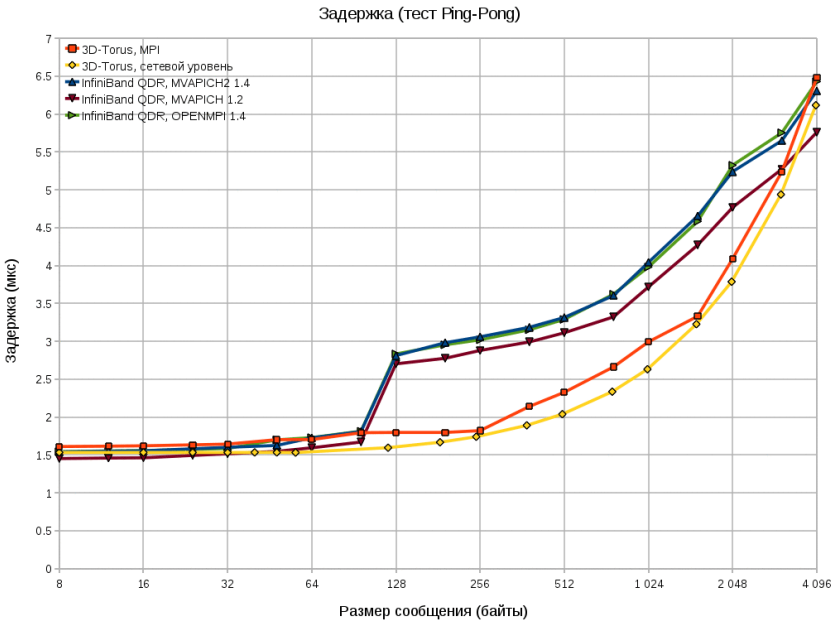

Рис. 4. Задержка (меньше — лучше)

– Сеть 3D-тор [3], реализованная авторами данной статьи в ПЛИС, имеющая 6 линков с пиковой пропускной способностью по 10 Гбит/с каждый.
– Сеть InfiniBand QDR с пиковой пропускной способностью 40 Гбит/с.

Нужно отметить, что в данных тестах использовался лишь один полнодуплексный линк 3D-тора, пиковая пропускная способность которого в 4 раза меньше чем у InfiniBand QDR. Тем не менее, видно, что нам удалось добиться существенного преимущества над сетью InfiniBand по темпу выдачи сообщений.

Ниже будет дано описание технического устройства нашей сети и будут показаны некоторые приемы, которые позволили достичь такого результата.

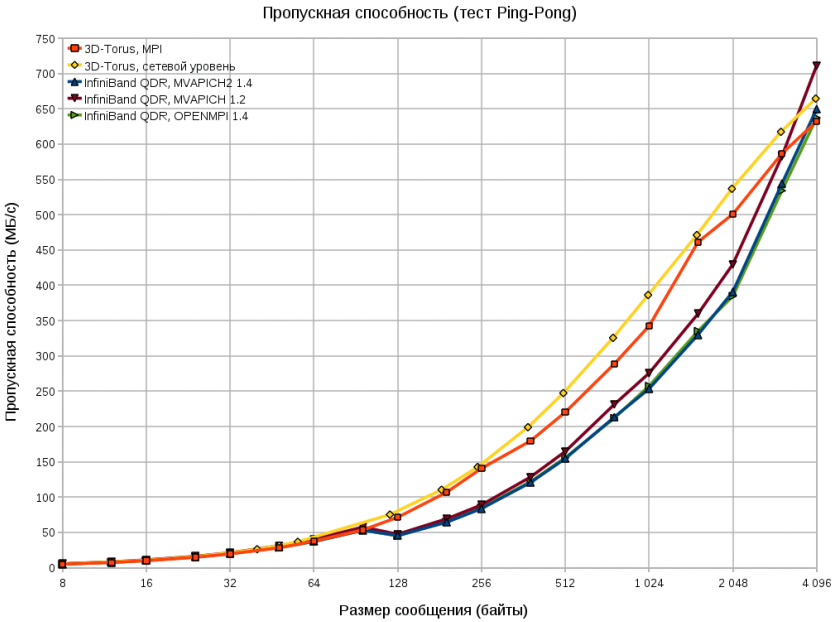

Рис. 5. Пропускная способность на тесте PingPong (больше — лучше)

3. Некоторые детали реализации сети 3D-тор суперкомпьютера «СКИФ-Аврора»

В качестве сетевого адаптера используется ПЛИС Stratix IV фирмы Altera [15] со специальной прошивкой, разработанной авторами данной статьи. Эта ПЛИС имеет:

- Встроенные ядра PCI Express в количестве двух штук.
- Набор высокоскоростных (по 10 Гбит/с) каналов, которые выходят на трансиверы и далее связываются в сеть с топологией трехмерного тора.
- Достаточное количество вентилей и статической памяти, чтобы реализовать маршрутизатор. Следует отметить, что сам маршрутизатор отнимает лишь небольшую долю этих ресурсов, так что есть возможность добавлять другое оборудование, например, специализированные под задачу вычислители.

Использование ПЛИС имеет ряд преимуществ перед созданием собственных интегральных схем. Прежде всего, это возможность инкрементальной разработки, специализации под задачу, реализация ускорителей [5]. Кроме того, собственные интегральные схемы имели бы высокую цену производства и более высокую вероятность ошибки. Последнее обусловлено тем, что маршрутизатор может быть хорошо отлажен только при наличии сети из достаточно большого количества узлов.

3.1. Топология сети

Идея использовать тороидальную топологию, в том числе трехмерную, не нова. В качестве примера можно отметить разработки Cray T3E [10] и IBM BlueGene/L [7].

Особенностью сети суперкомпьютера «СКИФ-Аврора» является возможность ее переконфигурации. А именно, можно перекоммутировать линки так, что всё множество узлов разбивается на несколько независимых подсетей, каждая из которых имеет топологию трехмерного тора. Этот механизм может иметь практическое значение при эксплуатации машины; он дает возможность изолировать запущенные на суперкомпьютере задачи друг от друга.

3.2. Маршрутизация

Задача построения маршрутизатора не так проста, как может показаться на первый взгляд. Неправильно сконструированная сеть может обладать такими неприятными свойствами, как, например, подверженность дедлокам или лайвлокам.

Дедлоком (англ. deadlock) называют ситуацию, при которой группа пакетов в сети не может продолжить движение из-за ожидания освобождения ресурсов, занятых самими этими пакетами. Если такая ситуация однажды возникла, то вся сеть или ее часть теряет способность передавать данные.

Существует два способа обойти проблему дедлоков:

(1) Выявлять дедлокоопасные ситуации и применять меры по их устранению.

(2) Использовать такую логику маршрутизации, при которой возникновение дедлока невозможно принципиально.

Наша сеть реализована по второму способу; в ней применяется принцип пузырьковой маршрутизации [9], который заключается в том, что пакету запрещено продолжать движение, если на приемной стороне нет так называемого «пузырька» — свободного места в буфере. Кроме того, накладываются некоторые ограничения на выбор пути пакета. Подробно все эти правила описаны, например, в работе Х. Дуато [8].

Лайвлок (англ. livelock) — это ситуация, при которой пакет бесконечно долго блуждает в сети, не находя узла-получателя. Бороться с ними довольно просто — маршруты в нашей сети выбираются только минимальной длины. Тогда при движении пакета расстояние до узла-получателя будет всегда сокращаться, и лайвлок будет невозможен.

Для повышения эффективности сети используется принцип *червячной маршрутизации*, описанный в [8]. Основная идея заключается в том, чтобы не дожидаться прихода конца пакета в промежуточном узле, а сразу, по приходу заголовка, передавать пакет дальше. Такое спекулятивное поведение дает значительный выигрыш в задержке, а также косвенно сказывается на пропускной способности сети. Конечно, реализация червячного принципа требует некоторой аккуратности — нужно позаботиться об обработке исключительных случаев, когда пакет обрывается или его содержимое портится и перестает удовлетворять контрольной сумме, — но игра определенно стоит свеч.

3.3. Стек протоколов

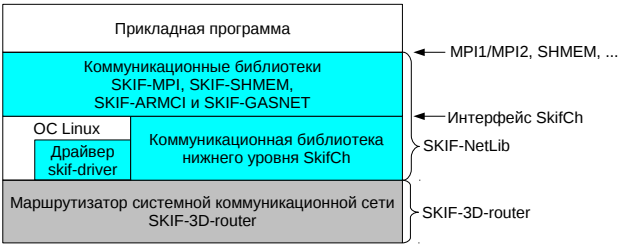

Рис. 6. Стек протоколов

На рис. 6 представлен стек протоколов передачи данных в сети суперкомпьютера «СКИФ-Аврора». Важная роль отводится интерфейсу SkifCh (он проходит примерно посередине схемы), который предоставляет возможность реализации любой коммуникационной библиотеки верхнего уровня. Авторами данной статьи были, в частности, реализованы библиотеки SKIF-MPI и SKIF-SHMEM, которые соответственно предоставляют MPI-2 [17] и SHMEM [18].

Программно-аппаратная реализация интерфейса SkifCh является одним из ключевых элементов сети «СКИФ-Аврора». Она обеспечивает передачу данных от пользовательского процесса в ПЛИС и обратно, а также обмен на общей памяти между процессами одного узла. Можно сказать, что именно в этом месте содержится ряд решений, позволивших обойти InfiniBand по темпу выдачи сообщений. Среди этих решений стоит упомянуть следующие:

- Облегченный по сравнению с InfiniBand формат сообщения. В результате уменьшена доля служебной информации в сообщении.
- Отсутствие уведомлений об отправке данных от процесса в сеть и обратно. В частности, факт записи процессом данных является достаточным для того, чтобы ПЛИС «догадалась», что пришло очередное сообщение.

Как видно на графиках (рис. 2, 3, 4, 5) интерфейс SkifCh (линия «3D-Torus, сетевой уровень») показывает заметно более высокие результаты, чем MPI. Это обусловлено накладными расходами на обработку сообщений в библиотеке MPI. Интерфейс SkifCh доступен программисту, позволяя ему воспользоваться всеми возможностями сети. В отличие от MPI, реализация интерфейса SHMEM является более «легкой», поэтому производительность программы на SHMEM близка к производительности программы с использованием интерфейса SkifCh.

4. Заключение

Наш опыт реализации коммуникационной сети собственной разработки для суперкомпьютера «СКИФ-Аврора» говорит о том, что

технологический прогресс в области стандартизации коммуникационных интерфейсов (таких как PCI-Express) и изготовления микросхем — в частности, схем с программируемой логикой (ПЛИС) — достиг такой стадии, когда эксперименты в области создания собственных сетевых решений оправданы и даже необходимы с точки зрения конкурентоспособности.

По большому счету, прежде всего сеть отличает один суперкомпьютер от другого. Точно так же, как сейчас имеются широкие возможности по построению гибридных машин с использованием всевозможных ускорителей — от графических до самостоятельно реализованных в тех же ПЛИС — на сегодняшний день есть реальная возможность строить специализированные сетевые «ускорители».

Мы показали, что используя аккуратно выверенные на всех уровнях протоколы передачи данных, можно построить решение, оставляющее далеко позади InfiniBand даже по такому важному в широком классе задач параметру, как темп выдачи сообщений, при этом оставаясь на том же уровне по величине коммуникационной задержки.

В заключение авторы хотели бы поблагодарить С. М. Абрамова за активную поддержку на всех стадиях выполнения проекта и А. О. Лациса за бесценные консультации.

Список литературы

[1] Абрамов С. М., Заднепровский В. Ф., Шмелев А. Б., Московский А. А. *Супер ЭВМ ряда 4 семейства СКИФ: штурм вершины суперкомпьютерных технологий* // Параллельные вычислительные технологии : Труды Международной научной конференции (30 марта – 3 апреля 2009 г., г. Нижний Новгород). — Нижний Новгород : Изд. Нижегородского государственного университета им. Н.И. Лобачевского, 2009, с. 5–16. ↑[]

[2] Климов Ю. А., Орлов А. Ю., Шворин А. Б. *Темп выдачи сообщений как мера качества коммуникационной сети* // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи : Труды Международной суперкомпьютерной конференции (20–25 сентября 2010 г., г. Новороссийск). — Москва : Изд-во МГУ, 2010, с. 414–417. ↑

[3] Орлов А. Ю., Шворин А. Б. *О реализации в ПЛИС маршрутизатора высокопроизводительной сети* // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность : Труды Всероссийской суперкомпьютерной конференции (21–26 сентября 2009 г., г. Новороссийск). — Москва : Изд-во МГУ, 2009, с. 208–210. ↑2.3

[4] Лацис А. О. Вычислительная система МВС-Экспресс, http://www.kiam.ru/MVS/research/mvs_express.html. ↑1

[5] Андреев С. С., Дбар С. А., Лацис А. О., Плоткина Е. А. *Система программирования Автокод HDL и опыт ее применения для схемной реализации численных методов в FPGA* // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность : Труды Всероссийской суперкомпьютерной конференции (21–26 сентября 2009 г., г. Новороссийск). — Москва : Изд-во МГУ, 2009, с. 237. ↑3

[6] Корж А. А. *Результаты масштабирования бенчмарка NPB UA на тысячи ядер суперкомпьютера Blue Gene/P с помощью PGAS-расширения OpenMP* // Вычислительные методы и программирование, 2010, № 11, с. 31–41. ↑2.1

[7] Adiga N. R., Blumrich M. A., Chen D., Coteus P., Gara A., Giampapa M. E., Heidelberger P., Singh S., Steinmacher-Burow B. D., Takken T., Tsao M., Vranas P. *Blue Gene/L Torus Interconnection Network* // IBM J. Research and Development, 2005. ↑3.1

[8] Duato J. *A necessary and sufficient condition for deadlock-free routing in wormhole networks* // IEEE Transactions on Parallel and Distributed Systems, 1995. **6**, p. 1055–1067. ↑3.2

[9] Puente V., Izuy C., Beivide R., Gregorio J. A., Vallejo F., Prellezo J. M. *The Adaptive Bubble Router* // Journal of Parallel and Distributed Computing, 2001. **61**, no. 9, p. 1180–1208. ↑3.2

[10] Scott S. L., Thorson G. M. *The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus* // HOT Interconnects IV : Stanford University, 1996. ↑3.1

[11] Набор тестов Intel MPI Benchmarks (IMB), http://software.intel.com/en-us/articles/intel-mpi-benchmarks/. ↑1, 1

[12] Набор тестов HPC Challenge Benchmark, http://icl.cs.utk.edu/hpcc/. ↑2.1

[13] Набор тестов NAS Parallel Benchmarks (NPB), http://www.nas.nasa.gov/Resources/Software/npb.html. ↑1, 2.1

[14] Тест Bandwidth, http://botik.ru/~klimov/bandwidth.tgz. ↑2.2

[15] Altera Stratix IV FPGA: transceiver overview, http://www.altera.com/products/devices/stratix-fpgas/stratix-iv/transceivers/stxiv-transceivers.html. ↑3

[16] Berkeley Open Infrastructure for Network Computing, http://boinc.berkeley.edu/. ↑1

[17] Message Passing Interface (MPI), http://www.mpi-forum.org/. ↑[], 2.1, 3.3

[18] SHMEM application programming interface, http://www.shmem.org/. ↑[], 2.1, 3.3

I. A. Adamovich, A. V. Klimov, Yu. A. Klimov, A. Y. Orlov, A. B. Shvorin. *Thoughts on the Development of SKIF-Aurora Supercomputer Interconnect.*

Abstract. This article presents speculations on the experience authors have got while developing the 3D-torus interconnect of SKIF-Aurora supercomputer. Authors have implemented all the levels of network infrastructure from the schematics of network adapters up to the support of communication libraries. Some advantages of self-made networks over the commercially available off-the-shelf solutions are discussed. We show that the last ones can be significantly surpassed in some aspects by careful engineering and implementation. As an example we compare message rate characteristics in our network and in InfiniBand.

Key Words and Phrases: supercomputers, HPC, network, interconnect, FPGA, routing, 3D-torus.

Поступила в редакцию 29.09.2010. *Образец ссылки на статью:*

И. А. Адамович, А. В. Климов, Ю. А. Климов, А. Ю. Орлов, А. Б. Шворин. *Опыт разработки коммуникационной сети суперкомпьютера «СКИФ-Аврора»* // Программные системы: теория и приложения : электрон. научн. журн. 2010. № 3(3), с. 107–123. URL: http://psta.psiras.ru/read/psta2010_3_107-123.pdf (дата обращения: 10.10.2010)