Анализ трафика Infiniband для построения коммуникационного профиля приложений*

А.А. Градсков[1], К.С. Стефанов[2]

Факультет вычислительной математики и кибернетики МГУ имени М.В.Ломоносова[1],
Научно-исследовательский вычислительный центр МГУ имени М.В.Ломоносова[2]

В данной статье представлен метод построения коммуникационной матрицы приложения путем анализа трафика, проходящего через сеть Infiniband. Предлагаемый метод основан на взаимодействии с элементами коммуникационной сети и не требует вмешательства в исходный код и работу программы. Во время работы приложения собирается его трафик, из которого извлекается информация о коммуникационном профиле. Реализация метода протестирована на NAS Parallel Benchmarks (NPB).

Ключевые слова: коммуникационный профиль, Infiniband, трафик

1. Введение

На производительность приложений, выполняющихся на суперкомпьютерах, сильное влияние оказывают задержки при передаче данных в коммуникационной сети. Поэтому важным является вопрос эффективного использования имеющейся сети. Для понимания того, какие есть возможности повышения эффективности, необходимо знать, каким образом реальные приложения используют коммуникационную сеть.

На данный момент существует несколько подходов к изучению поведения приложений в коммуникационной сети. Один из них — добавление в исходный код приложения специальных инструкций, которые будут записывать его взаимодействие с сетью. Этот метод требует непосредственного изменения исходного кода приложения. Это не всегда возможно и может быть трудозатратно.

Второй метод — использование встроенных возможностей библиотек. К инструментам такого вида можно отнести PMPI [4] — стандартный интерфейс профилирования MPI-приложений. Он позволяет замещать вызовы функций MPI обертками с необходимым дополнительным поведением. Этот интерфейс могут использовать сторонние библиотеки анализа выполнения программ, например: Kojak [6], Scalasca [5], Expert [6]. Они позволяют проводить трассировку приложения и анализ трасс. Подобные инструменты требуют проведения инструментирования программы, поэтому им также необходим доступ к исходному коду.

Также к инструментам второго вида относится библиотека mpiP [7]. Она предоставляет возможность проводить статистический анализ коммуникаций MPI приложений. Такой подход позволяет уменьшить накладные расходы на профилирование.

В данной работе предлагается метод сбора коммуникационного профиля приложения. Результат представляется в виде коммуникационной матрицы, в которой для каждой упорядоченной пары процессов (отправителя и получателя) приложения записано количество информации, переданной от отправителя получателю за время выполнения программы. Сбор данных не требует модификации приложения или его перекомпиляции и предполагает использование инструментов сети Infiniband, имеющихся для коммутаторов и сетевых карт компании Mellanox [8].

В разделе 2 описываются элементы архитектуры Infiniband, затрагиваемые в работе.

*Работа выполняется при финансовой поддержке РФФИ, грант 16-07-01121. Работа выполнена с использованием ресурсов суперкомпьютерного комплекса МГУ имени М.В.Ломоносова.

Далее (раздел 3) представлен набор использованных инструментов. В разделе 4 излагается разработанный метод. Его тестирование описано в разделе 5.

2. Особенности сети Infiniband

Сеть Infiniband делится на подсети, соединенные между собой маршрутизаторами. Подсети могут состоять из коммутаторов и узлов сети, соединенных между собой. У каждого вычислительного узла сети имеется Host Channel Adapter (HCA-адаптер), с помощью которого производятся коммуникации.

У каждой подсети существует subnet manager (SM). С его помощью сеть настраивается перед использованием. В частности, SM присваивает каждому элементу подсети (маршрутизаторам, коммутаторам, узлам) уникальный идентификатор – Local Identifier (LID). Он используется для адресации внутри подсети. Также SM определяет возможные пути коммуникации между узлами и для каждой пары LID фиксирует путь, который будет в дальнейшем использоваться для передачи данных между ними.

При отправке пакета по сети отправитель должен задать поле SLID (source LID) в заголовке пакета (Local Route Header, LRH) равным своему LID, а поле DLID (destination LID) равным LID получателя пакета.

3. Используемые инструменты

Для Infiniband существует аналог утилиты tcpdump [3] — ibdump. Она позволяет перехватывать пакеты Infiniband, приходящие на узел вычислительной сети и сохранять их в файл в бинарном формате для дальнейшего анализа. Для работы инструмента требуется наличие на узле HCA-адаптера, с которым он будет взаимодействовать.

Для дублирования и перенаправления трафика на коммутаторах сети используется утилита ibmirror. Она позволяет продублировать входящий и/или исходящий трафик некоторых портов коммутатора на заданный порт.

Утилита ibtracert позволяет получить путь в сети, по которому будет проходить пакет, между некоторыми двумя точками сети. Получателя и отправителя можно задать с помощью LID.

В качестве планировщика задач использовался планировщик задач slurm [1]. Он позволяет перед запуском приложения на одном из выделенных им узлов выполнить скрипт, в котором производится настройка. Далее описываются действия, выполняемые использованным скриптом.

4. Предлагаемый метод

4.1. Общее описание

Предполагаемый метод позволяет построить коммуникационную матрицу приложения путем сбора и анализа трафика. Во время работы приложения сетевые пакеты, проходящие в коммуникационной сети суперкомпьютера между задействованными вычислительными узлами, перехватываются и сохраняются на диск. Затем из них извлекается информация о отправителе и получателе, времени и размере сообщения. По этим данным уже можно непосредственно построить коммуникационную матрицу.

4.2. Получение трасс

Для получения трафика во время работы приложения используется инструмент ibdump. Он запускается на одном из вычислительных узлов сети и перехватывает входящий трафик, предназначенный этому узлу. Таким образом, чтобы получить информацию о всем трафике,

проходящем по сети во время работы приложения, используя только ibdump, понадобилось бы запустить его на каждом вычислительном узле, на котором работает исследуемое приложение.

Для снижения влияния на работу приложения трафик можно перенаправить (продублировать) на выделенный узел (возможно, несколько узлов) сети, не задействованный в вычислениях, на котором централизованно снимать данные. Перенаправление настраивается на каждом коммутаторе по-отдельности. Для каждого их них выделяется узел, не принимающий участие в вычислениях, и трафик перенаправляется на него. Там сетевые пакеты перехватываются через ibdump.

Для соответствующей настройки сети и запуска утилит требуется знать следующее:

- набор узлов, на которых будет запускаться приложение;
- коммутаторы, которые будут использоваться для передачи данных;
- узлы, которые можно использовать для сбора трафика.

Эту информацию можно получить, используя планировщик задач slurm. Команда srun позволяет выполнить переданное ей задание на всех выделенных планировщиком узлах. С ее помощью можно от каждого узла получить нужную информацию. Через srun на каждом узле выполняется команда hostname, которая показывает имя узла в вычислительной сети, необходимое для запуска MPI-приложения и подключения к узлу для дальнейшей настройки. С помощью утилиты ibstat для каждого узла узнается его LID в сети. Имена узлов и соответствующие им LID сохраняются в файл для дальнейшей работы.

Далее происходит получение информации о используемых коммутаторах. Для этого используется команда ibtracert — она позволяет проследить путь между двумя узлами сети, задаваемыми через LID. Команда запускается для каждого узла, при этом в качестве отправителя указывается этот узел, а в качестве получателя — узел, на котором выполняется скрипт. В выдаче ibtracert показываются конечные узлы, коммутаторы и их порты, использованные для связи между отправителем и получателем. Из этой информации для каждого узла выделяется коммутатор и его порт, к которому непосредственно подключен данный узел.

Для каждого найденного коммутатора выделяется один из узлов, подключенных к нему. Этот узел будет использоваться для перехвата трафика, проходящего через данный коммутатор. На коммутаторе происходит настройка перенаправления трафика — все входящие и исходящие пакеты каждого из портов, к которым подключены вычислительные узлы, дублируются на порт выделенного узла. На этом узле запускается ibdump.

Из свободных вычислительных узлов формируется конфигурационный hostfile, который используется для запуска исследуемого приложения с помощью mpirun.

После окончания работы приложения запущенные на узлах процессы ibdump завершаются, сохраняя перехваченный трафик в файл. На коммутаторах сбрасывается конфигурация перенаправления.

На рисунке 1 графически представлен пример распределения узлов сети во время работы приложения. В работе участвуют два коммутатора, к каждому из которых подключены по четыре узла сети. Для каждого коммутатора выделен узел сброса, который принимает данные, проходящие через все задействованные порты коммутатора.

4.3. Обработка трафика

Построение коммуникационного профиля требует информации о передаче данных между каждой парой вычислительных узлов во время работы приложения. Эта информация извлекается из копии трафика сети Infiniband, полученной описанным способом.

В собранном трафике, кроме данных, передаваемых приложением, встречаются системные пакеты. Такие пакеты определяются по их типу, записанному в заголовке пакета, и не

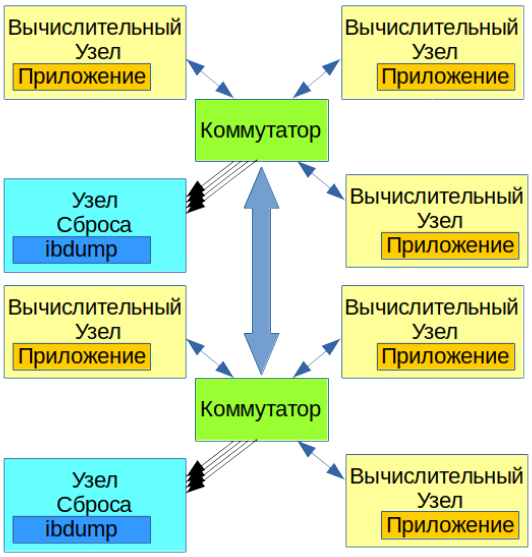

Рис. 1. Схема сбора данных во время работы приложения

рассматриваются далее.

Для каждого пакета из его заголовка LRH извлекаются LID отправителя и получателя, а также время получения пакета. Используя эту информацию, все пакеты размещаются в таблице, где строки соответствуют отправителю, а столбцы — получателю. Используя данную таблицу легко получить коммуникационную матрицу приложения — достаточно для каждой ячейки сложить размеры всех пакетов в ней.

Для полного исследования трафика приложения необходимо, чтобы процессы были распределены по одному на вычислительный узел. Если это не так, то указанных в заголовке пакета LID будет недостаточно для классификации по отправителю и получателю. Поэтому получаемые данные коммуникационного профиля будут относится к группам процессов, расположенных на одном узле. Чтобы частично преодолеть это ограничение, можно использовать MPI-ранги процессов. Чтобы перейти от физических узлов сети к логическим номерам (рангам), необходимо провести соответствие LID узлов рангам процессов, выполняющихся на них. Для этого можно использовать содержимое определенных пакетов MPI-приложения. В первом пакете каждого MPI-сообщения содержится ранг отправителя сообщения, который можно соотнести с LID вычислительного узла. С помощью этой информации можно получить отображение между LID узлов и MPI-рангами выполняющихся на узле процессов.

5. Тестирование

Тестирование производилось на суперкомпьютере Ломоносов [2].

Работа метода была проверена для некоторых тестов из набора NAS Parallel Benchmarks (NPB).

На рисунке 2 графически представлена коммуникационная матрица для теста BT из NPB (Tri-Diagonal Solver, параметры CLASS=S, NPROCS=9). Вычисления происходят на трехмерной сетке размером 12x12x12. Приложение выполнялось на девяти узлах, еще на трех узлах производился сбор данных. Для строк и столбцов указаны LID отправителя и получателя, а цветом указан размер переданных во время работы данных. На цветовой шкале показан соответствующий размер в килобайтах.

Для теста BT было проведено сравнение со средством инструментирования Score-P.

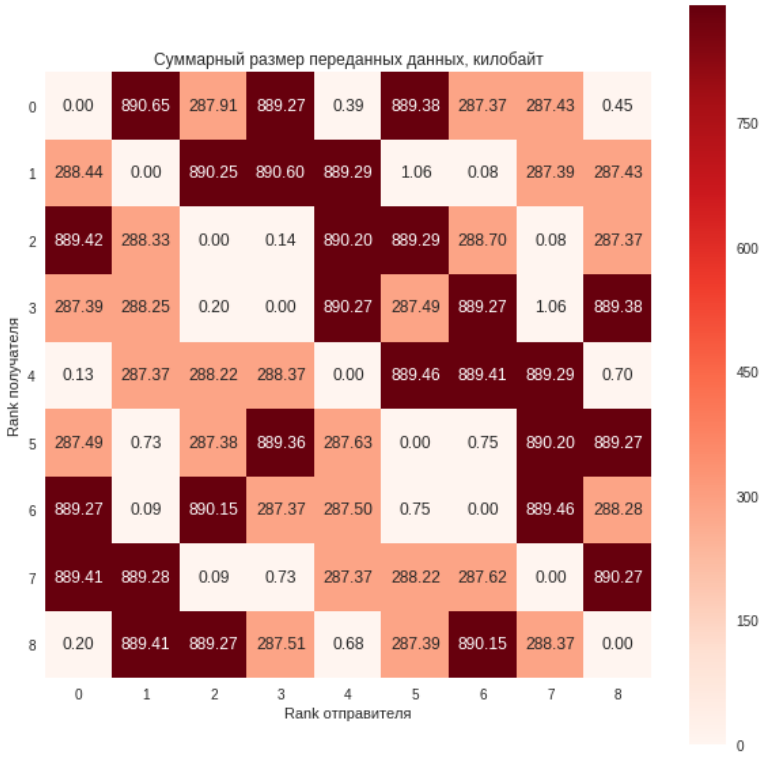

Рис. 2. Коммуникационная матрица для теста BT в NPB

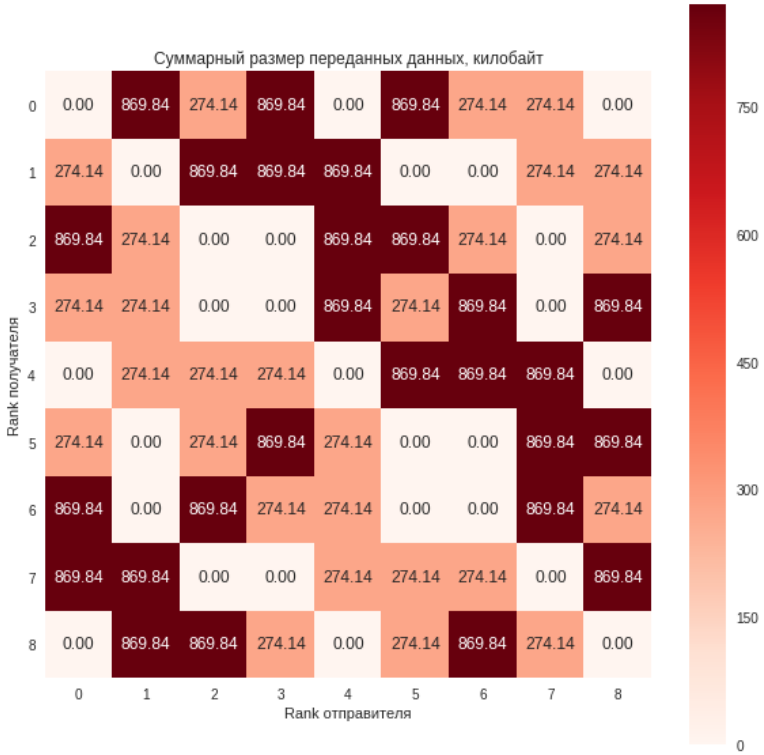

Рис. 3. Коммуникационная матрица для теста BT в NPB, полученная с помощью Score-P и VAMPIR

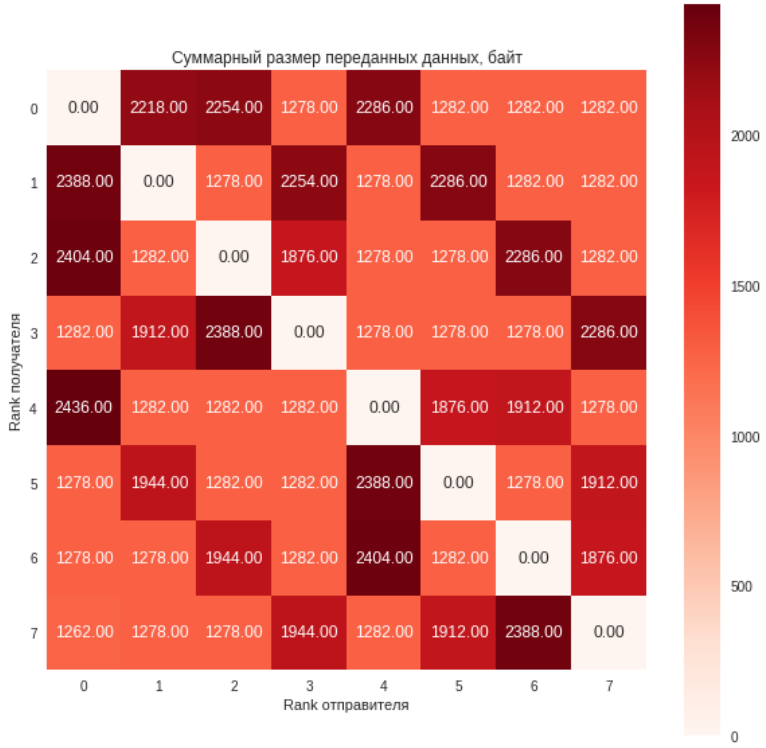

Рис. 4. Коммуникационная матрица для теста FT в NPB

На рисунке 3 представлена коммуникационная матрица, полученная инструментом VAMPIR [9] из трассы приложения, собранной Score-P. Размер зарегистрированных данных отличается не более чем на 6%, что объясняется учетом заголовков пакетов Infiniband.

На рисунке 4 графически представлена коммуникационная матрица приложения, вычисляющего быстрое преобразование Фурье в трехмерном пространстве — тест FT в наборе NPB (параметры CLASS=S, NPROCS=8). В данной конфигурации сетка имеет размер 64x64x64 и производится 6 итераций алгоритма. Приложение выполнялось на восьми узлах, еще на трех узлах производился сбор данных. Для данного теста не приводится сравнение с VAMPIR, так как FT использует только коллективные операции, которые не отображаются в матрицах VAMPIR. На рисунке же представлены фактические пересылки данных, которыми реализуются коллективные операции.

6. Заключение

В данной работе получен метод, позволяющий построить коммуникационную матрицу приложения по его трафику во время выполнения на вычислительной сети. Он позволяет исследовать приложение, не изменяя его исходный код и не пересобирая его.

Литература

1. Yoo A. B., Jette M. A., Grondona M. Slurm: Simple linux utility for resource management //Workshop on Job Scheduling Strategies for Parallel Processing. – Springer Berlin Heidelberg, 2003. – С. 44-60.

2. Воеводин Вл.В., Жуматий С.А., Соболев С.И., Антонов А.С., Брызгалов П.А., Никитенко Д.А., Стефанов К.С., Воеводин Вад.В. Практика суперкомпьютера "Ломоносов"// Открытые системы. - Москва: Издательский дом "Открытые

системы"N 7, 2012. С. 36-39.

3. Jacobson V., Leres C., McCanne S. The tcpdump manual page //Lawrence Berkeley Laboratory, Berkeley, CA. – 1989. – Т. 143.
4. Karrels E., Lusk E. Performance analysis of MPI programs //Proceedings of the Workshop on Environments and Tools For Parallel Scientific Computing. – 1994. – С. 195-200.
5. Geimer M. et al. The Scalasca performance toolset architecture //Concurrency and Computation: Practice and Experience. – 2010. – Т. 22. – №. 6. – С. 702-719.
6. Mohr B., Wolf F. KOJAK–A tool set for automatic performance analysis of parallel programs //European Conference on Parallel Processing. – Springer Berlin Heidelberg, 2003. – С. 1301-1304.
7. Vetter J., Chambreau C. mpip: Lightweight, scalable mpi profiling. – 2005.
8. Mellanox Technologies [online] Available: http://www.mellanox.com.
9. Nagel W. E. et al. VAMPIR: Visualization and analysis of MPI resources. – 1996.

Infiniband traffic analysis for communication profile construction

A.A. Gradskov[1], K.S. Stefanov[2]

Faculty of Computational Mathematics and Cybernetics
of Lomonosov Moscow State University[1],
Research Computing Center M.V. Lomonosov Moscow State University[2]

This article presents a method of constructing a communication matrix of the application by analyzing the traffic through the Infiniband network. The proposed method is based on the interaction with elements of the communication network and does not require intervention into the source code and the program execution. During the operation of the application its traffic is gathered, from which information about the communication profile is extracted. Implementation of the method is tested on NAS Parallel Benchmarks.

Keywords: communication profile, Infiniband, traffic

References

1. Yoo A. B., Jette M. A., Grondona M. Slurm: Simple linux utility for resource management //Workshop on Job Scheduling Strategies for Parallel Processing. – Springer Berlin Heidelberg, 2003. – С. 44-60.

2. Voevodin Vl.V., Zhumatij S.A., Sobolev S.I., Antonov A.S., Bryzgalov P.A., Nikitenko D.A., Stefanov K.S., Voevodin Vad.V. Praktika superkomputera "Lomonosov"// Otkrytye Sistemy [Practice of “Lomonosov” supercomputer //Open Systems J. – 2012. – No. 7. – P. 36-39.]

3. Jacobson V., Leres C., McCanne S. The tcpdump manual page //Lawrence Berkeley Laboratory, Berkeley, CA. – 1989. – Т. 143.

4. Karrels E., Lusk E. Performance analysis of MPI programs //Proceedings of the Workshop on Environments and Tools For Parallel Scientific Computing. – 1994. – С. 195-200.

5. Geimer M. et al. The Scalasca performance toolset architecture //Concurrency and Computation: Practice and Experience. – 2010. – Т. 22. – No. 6. – С. 702-719.

6. Mohr B., Wolf F. KOJAK–A tool set for automatic performance analysis of parallel programs //European Conference on Parallel Processing. – Springer Berlin Heidelberg, 2003. – С. 1301-1304.

7. Vetter J., Chambreau C. mpip: Lightweight, scalable mpi profiling. – 2005.

8. Mellanox Technologies [online] Available: http://www.mellanox.com.

9. Nagel W. E. et al. VAMPIR: Visualization and analysis of MPI resources. – 1996.