МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра: МОСТ**

Направление подготовки: «Фундаментальная информатика и информационные технологии»

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**Оптимальные коммуникационные взаимодействия в высокопроизводительных вычислениях**

**Выполнил**:

студент группы 381706-1м

Гладков Дмитрий Олегович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

**Научный руководитель**:

Доцент каф. МОСТ, к.т.н.,

Сысоев Александр Владимирович

ученая степень, ученое звание, ф.и.о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Нижний Новгород  
2017

Содержание

[Аннотация 3](#_Toc502310075)

[Введение 4](#_Toc502310076)

[Цель работы 7](#_Toc502310077)

[Обзор предметной области 8](#_Toc502310078)

[Заключение 13](#_Toc502310079)

[Литература 14](#_Toc502310080)

Аннотация

За последние десятилетия были разработаны и представлены несколько высокоскоростных сетей, каждая из которых отличается от других показателями производительности, коммуникационными особенностями и возможностями. Однако инициативы по разработке новых сетевых решений сопровождаются увеличением расхождений в коммуникационных интерфейсах или “языках”, используемых каждой сетью. Соответственно, переносимость приложений на различных коммуникационных интерфейсах стала темой обширных исследований. В качестве основного средства для достижения переносимости приложений широко использовались такие модели программирования, как Sockets, Shared Memory, Message Passing Interface (MPI), Symmetric Hierarchical MEMory (SHMEM), Global Arrays (GA), Charm++ и другие.

В данной работе исследуются различные подходы к созданию унифицированного программного интерфейса доступа к сети, которая может быть использована различными параллельными программными моделями, такими как MPI, SHMEM и другими, в различных высокоскоростных сетевых средах (например, InfiniBand, Omni-Path Architecture, Ethernet, а также решения, основанные на Ethernet – iWARP, RoCE). В частности, исследования в данной работе посвящены:

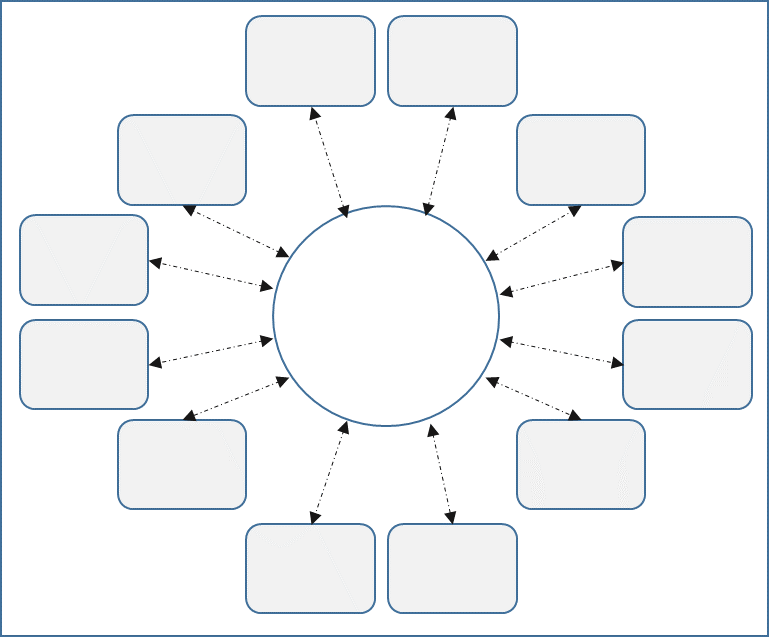
1. анализу существующих программных моделей для параллельного программирования и их требований к коммуникационным интерфейсам;
2. рассмотрению существующих и широкоиспользуемых программных интерфейсов для работы с высокоскоростными сетями и возможности построения единой программной модели, способной обеспечить переносимость приложений;
3. анализу ограничений перспективных коммуникационных интерфейсов для мультисетевого программирования, рассмотрению и разработке возможностей для обеспечения наибольшей эффективности сетевого стека для параллельных программных моделей.

Введение

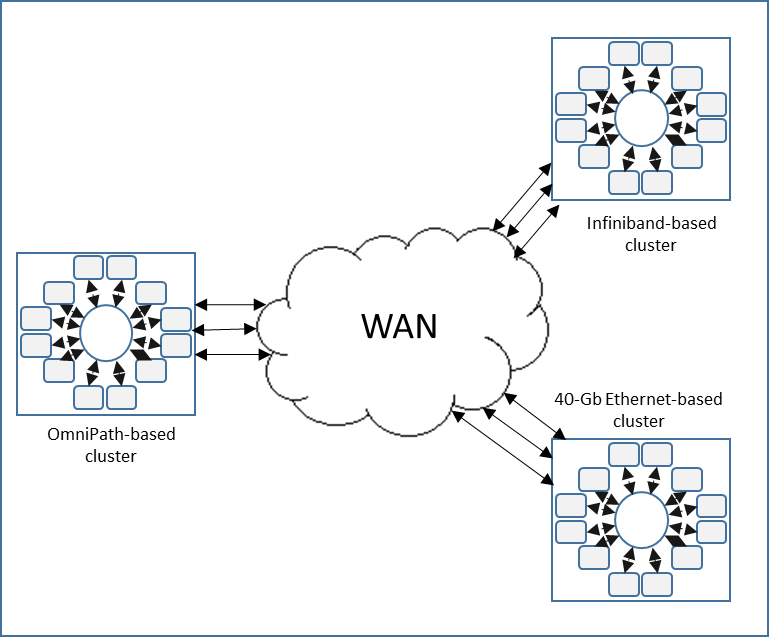
За последние несколько десятилетий достигнуты большие успехи в области компьютерных технологий. Компьютеры проникли во все области деятельности человека, будь то промышленность или исследования, и оказывают влияние на каждый аспект человеческой жизни. В начале 70-х гг XX века, когда обычные компьютеры набирали популярность среди исследователей, возникла необходимость в более мощных машинах, которые могли бы решить проблемы, слишком сложные для обычных компьютеров. Это осознание привело к развитию суперкомпьютеров – передовых и мощных машин, состоящих из нескольких обрабатывающих подсистем. Cray-1, разработанный Cray Research (ныне, Cray Inc.), был одним из таких мощных суперкомпьютеров. Развитие суперкомпьютеров и компьютерных технологий сопутствовало экспоненциальному росту требований приложений. Высокая стоимость проектирования, разработки и поддержки таких суперкомпьютеров для обеспечения высокой производительности приложений требовала, чтобы исследователи искали альтернативу этим суперкомпьютерам в виде кластерных систем (кластеров).

Кластеры состоят из недорогих коммерческих, готовых к использованию без дополнительных модификаций компьютеров (готовое коммерческое изделие, Commercial off-the-shell – COTS), соединенных между собой посредством сети. Кластеры становятся все более популярными в различных сферах применения, главным образом, из-за высокой производительности и относительно низкой стоимости. Такие системы теперь могут быть спроектированы для достижения различного уровня производительности из-за увеличения производительности процессоров, памяти и сетевых технологий.

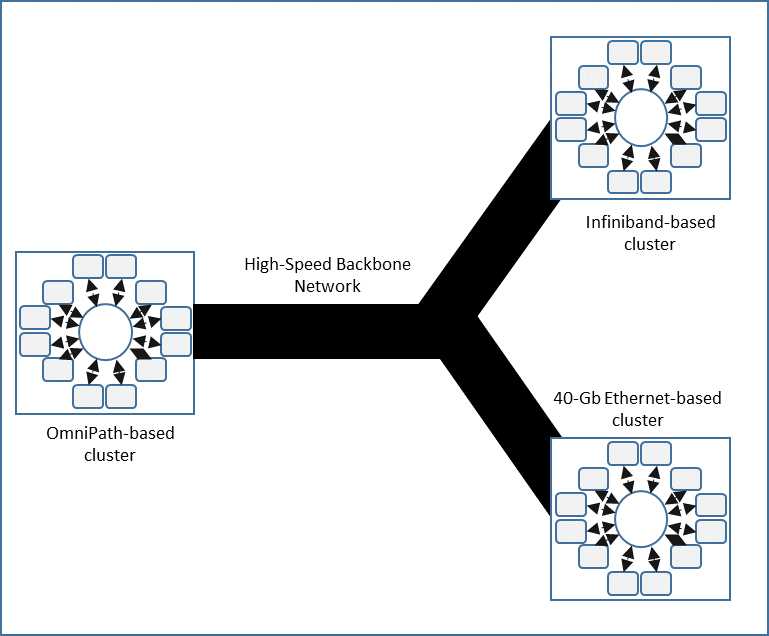
Поскольку системы на основе кластеров опираются на возможности вычислительных узлов, взаимодействующих друг с другом по сети, характеристики сети (аппаратное обеспечение, а также связанное с ним программное обеспечение) являются важным компонентом в эффективности и масштабируемости кластеров. На рис. 1 показаны типичные среды, используемые параллельными и распределенными вычислительными приложениями. Среды могут варьироваться от локальных сетей внутри одного кластера до нескольких различных кластеров, подключенных через WAN или высокоскоростную магистральную сеть.



(а)



(b)



(c)

Рис. 1. Типичные коммуникационные среды для распределенных и параллельных приложений:

(a) Связь внутри локальной сети кластера;

(b) Межкластерная связь по WAN;

(c) Межкластерная связь по высокоскоростной магистральной сети.

За последние несколько лет, на рынке был представлен ряд высокоскоростных сетей, включая 1/10/40/100-гигабитный Ethernet, InfiniBand, Myrinet, Omni-Path и т.д. С появлением таких сетей, коммуникационные издержки в кластерных системах переходят от самой сети к сетевым протоколам на стороне отправителя или получателя. Сетевые протоколы предыдущего поколения, такие как TCP/IP, полагаются на ядро операционной системы для обработки сообщений. Это вызывает множество копирований и переключений контекста ядра на критическом пути обработки сообщения. Таким образом, накладные расходы на связь были высокими. В течение последних нескольких лет исследователи рассматривают альтернативы для увеличения производительности коммуникаций в кластерах, посредством разработки протоколов низкого уровня с низкими задержками и высокой пропускной способностью, таких как FM и GM для Myrinet, EMP (Ethernet Message Passing) для Gigabit Ethernet и т.д. Чтобы стандартизировать эти решения, в конце 90-х годов была предложена архитектура виртуального интерфейса (VIA), но он не смог добиться успеха. В последние годы в индустрии была стандартизирована InfiniBand Architecture для разработки кластеров нового поколения. Все это приводит к сокращению отставания между функциональными возможностями физической сети и тем, что получают конечные пользователи.

Тем не менее, каждая новая сеть и протоколы пользовательского уровня предоставляют пользователю новые API (Application Programming Interface) или «язык» для взаимодействия с пользователем. Хотя эти новые API могут быть эффективно использованы для разработки новых приложений, они могут оказаться непригодными для уже существующих приложений, которые были разработаны несколько лет назад. Разработчики приложений нацелены на переносимость на различные будущие платформы и сети. Модель параллельного программирования MPI широко признана в качестве приемлемого подхода для достижений такой переносимости и является де-факто стандартом для научных приложений.

Для обеспечения переносимости основные реализации MPI основываются на широко используемом интерфейсе Sockets. Интерфейс Sockets является стандартом для написания сетевых приложений и поддерживается большим количеством современных системам, но не позволяет в полной мере использовать коммуникационные возможности сети. Каждая операция с пользовательскими данными несет значительные накладные расходы для обеспечения надежности, правильной последовательности, которые перекрывают низкие затраты сетевого интерфейса.

Цель работы

Имея ввиду поставленные проблемы Sockets интерфейса для построения высокоскоростных коммуникаций между объектами сети, которые не позволяют раскрыть полностью возможности сетевого оборудования, и проблемы специализированных API, разработанных для конкретных сетей, которые ограничивают переносимость приложений, в данном дипломном проекте предлагается рассмотреть перспективные подходы к созданию унифицированного интерфейса для различных параллельных технологий со следующими возможностями:

1. Позволять приложениям не только работать непосредственно в различных высокоскоростных сетях без каких-либо изменений, но также иметь возможность извлекать наилучшую производительность из сети.
2. Иметь возможность извлекать выгоду от предоставляемых способностей сетей, не ограничивая производительность и переносимость при переходе на другие сетевые технологии.
3. Обеспечивать надежность передачи данных, высокую масштабируемость приложений.

Цели данной работы состоят в том, чтобы:

1. Исследовать один из современных программных интерфейсов, который предоставляет высокопроизводительное, многофункциональное, переносимое промежуточное ПО для существующих параллельных моделей, используемых для высокопроизводительных вычислений, таких как MPI, GlobalArrays, SHMEM и другие.
2. Рассмотреть и предложить возможные решения для текущих недостающих функциональных возможностей исследуемого ПО, которые мешают обеспечивать низкие задержки, высокую пропускную способность, требуемую масштабируемость на известных сетевых технологиях.
3. Провести исследования, экспериментальные доказательства состоятельности и возможности дальнейшего применения предложенных решений на примере широко используемого программного интерфейса программирования высокопроизводительных приложений MPI.

Обзор предметной области

Последние десятки лет, мощность вычислительных узлов увеличивается в два раза каждые 18 месяцев. В тоже время современные высокоскоростные сети, которые обеспечивают высокую пропускную способность и низкие задержки также являются неотъемлемой частью для построения высокопроизводительных вычислительных сред. Как отмечалось ранее, современные высокопроизводительные системы создаются путем объединения вычислительных узлов посредством коммуникационных интерфейсов в кластерные системы. Кластерные системы зависят от вычислительных мощностей узлов и производительности сети. Таким образом развитие сетевых интерфейсов и ПО, поставляемого вместе с сетевыми интерфейсами, влияет на эффективность кластерных систем.

Как отмечалось ранее, большинство научных приложений использует MPI для достижения распределенности вычислений, поэтому в данной работе основные исследования связаны с MPI.

Стандарт MPI определяет четыре коммуникационных режима:

* Стандартный (Standard). В этом режиме решение о том, будет ли исходящее сообщение буферизовано или нет, принимает MРI. MРI может буферизовать исходящее сообщение. В таком случае операция посылки может завершиться до того, как будет вызван соответствующий прием. С другой стороны, буферное пространство может отсутствовать или MРI может отказаться от буферизации исходящего сообщения из-за ухудшения характеристик обмена. В этом случае операция отправки не будет завершена, пока данные не будут перемещены в процесс-получатель. Следовательно, в стандартном режиме посылка может стартовать вне зависимости от того, выполнен ли соответствующий прием. Она может быть завершена до окончания приема. Посылка в стандартном режиме является нелокальной операцией: она может зависеть от условий приема.
* Буферизованный (Buffered). Данный режим операции посылки может стартовать вне зависимости от того, инициирован ли соответствующий прием. Однако, в отличие от стандартной посылки, эта операция является локальной и ее завершение не зависит от обстоятельств приема. Следовательно, если посылка выполнена и никакого соответствующего приема не инициировано, тогда MPI обязан буферизовать исходящее сообщение, чтобы позволить завершиться соответствующему вызову операции отправки. Если не имеется достаточного объема буферного пространства, возникнет ошибка. Объем буферного пространства задается пользователем. Для того, чтобы буферизованный режим был эффективным, может потребоваться распределение буферов пользователем.
* Синхронный (Synchronous). Посылка, которая использует синхронный режим, может стартовать вне зависимости от того, был ли начат соответствующий прием. Однако, посылка будет завершена успешно, только если соответствующая операция приема стартовала. Следовательно, завершение синхронной передачи не только указывает, что буфер отправителя может быть повторно использован, но также и отмечает, что получатель достиг определенной точки в своей работе, а именно, что он начал выполнение приема. Если и посылка, и прием являются блокирующими операциями, тогда использование синхронного режима обеспечивает синхронную коммуникационную семантику: посылка не завершается на любой стороне обмена, пока оба процесса не выполнят синхронизацию (так называемый, рандеву) в процессе операции обмена. Выполнение обмена в этом режиме является нелокальным.
* По готовности (Ready). Посылка, которая использует режим обмена по готовности, может быть запущена только тогда, когда прием уже инициирован. В противном случае операция является ошибочной и результат будет неопределенным. На некоторых системах обмен по готовности позволяет устранить необходимость в синхронизации (рандеву), что улучшает характеристики обмена. Завершение операции посылки не зависит от состояния приема и в основном указывает, что буфер посылки может быть повторно использован. Операция посылки, которая использует режим по готовности, имеет ту же семантику, как и стандартная или синхронная передача. Это означает, что отправитель обеспечивает систему дополнительной информацией (а именно, что прием уже инициирован), которая может уменьшить накладные расходы.

Для разработки этих четырех коммуникационных режимов, обычно используются два внутренних протокола, Eager и Rendezvous. Эти протоколы обрабатываются механизмом процессинга (компонента в реализации MPI – Progress engine). Реализация Eager протокола обеспечивает передачу сообщений на удаленную (принимающую) сторону независимо от ее состояния. В Rendezvous протоколе, принимающая и отправляющая стороны участвуют в двухстороннем “рукопожатии” (handshake) через специальные сервисные сообщения перед тем, как данные будут переданы на принимающую сторону. Обычно, Eager протокол используется для маленьких сообщений (< 16 килобайт), а Rendezvous протокол используется для больших сообщений.

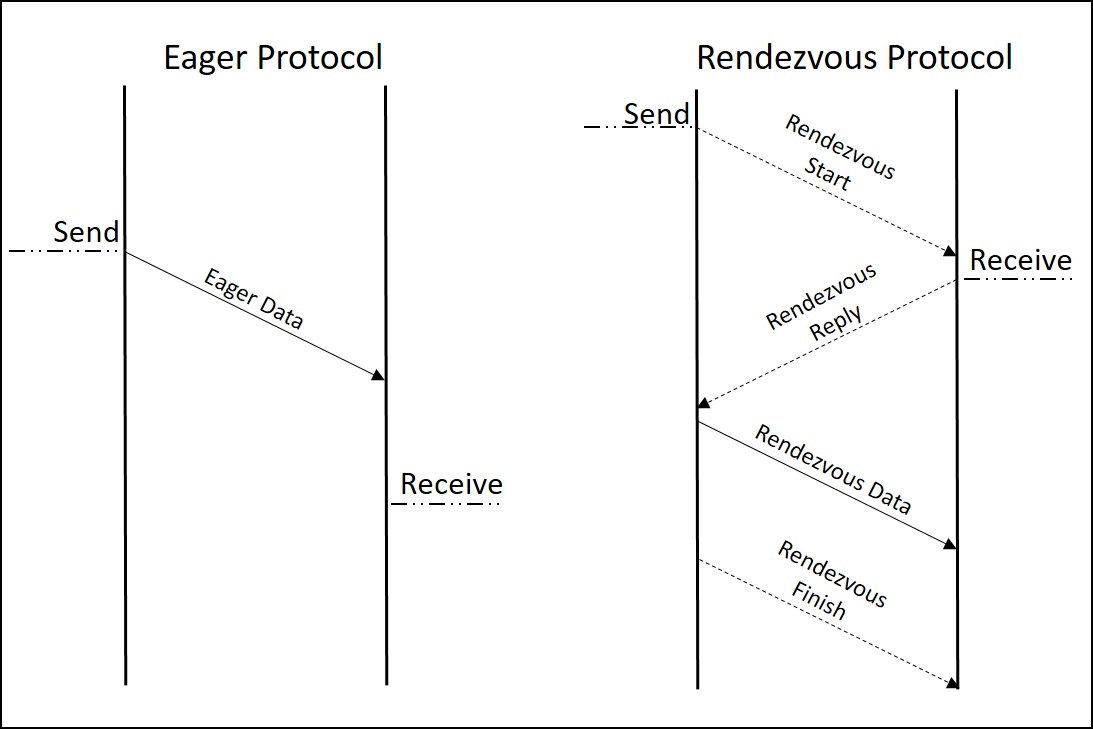


Рис. 2. Типичная реализация Eager и Rendezvous протоколов.

При передаче больших буферов с данными намного оптимальнее не выполнять дополнительных копирований данных, возникающих при использовании TCP/IP стека, который копирует пользовательские данные в ядро операционной системы и обратно.

InfiniBand Architecture (IBA) была предложена как следующее поколение для I/O и межпроцессных взаимодействий. InfiniBand[1] становится все более популярным для кластерных вычислений, благодаря открытому стандарту и высокой производительности. До недавнего времени параллельные технологии, такие как MPI, разрабатывались поверх Ethernet, с использованием широко используемого TCP/IP стека. Но с развитием IBA, стали появляться MPI реализации, которые способны работать поверх InfiniBand для обеспечения высокой производительности взаимодействия процессов, развернутых на разных узлах.

InfiniBand определяет сетевую фабрику для взаимодействия вычислительных и I/O узлов. Она предоставляет сетевую инфраструктуру и инфраструктуру управления для обеспечения межпроцессорного взаимодействия и I/O. В InfiniBand сети, вычислительные узлы и I/O узлы объединяются в фабрику посредством Channel Adapters (CA). Host Channel Adapter (HCA) располагаются на вычислительных узлах.

Коммуникационный стек InfiniBand состоит из нескольких уровней. Интерфейс, предоставляемый CAs потребителю, относится к транспортному уровню. В этом интерфейсе используется Queue-Pair модель. В InfiniBand Queue-Pair состоит из двух очередей: отправляющей и принимающей. Очередь отправки содержит инструкции для отправки данных, очередь приема содержит инструкции, которые описывают, где полученные данные должны быть расположены. Операции взаимодействия описываются в виде Work Queue Requests (WQR) и закладываются в отдельную очередь WQRs. Готовность операции, заложенной в очередь WQRs, доставляется с помощью очереди готовности, которая называется Completion Queue (CQ). InfiniBand поддерживает различные классы сервиса передачи данных – RC (Reliable Connections), Unreliable Datagram (UD), Unreliable Connections (UC).

Для обеспечения высокой производительности взаимодействия процессов IBA определила Remote Direct Memory Access (RDMA – удаленный прямой доступ к памяти), который обеспечивает передачу данных между узлами без вовлечения операционной системы и сетевого стека принимающей стороны в процесс передачи данных, позволяя минимизировать использование вычислительных ресурсов для получения данных из сети, тем самым освобождая процессорное время для вычислительной нагрузки. Данный механизм позволяет передавать большие буферы данных наиболее оптимальным образом, тем самым обеспечивая zero-copy передачу.

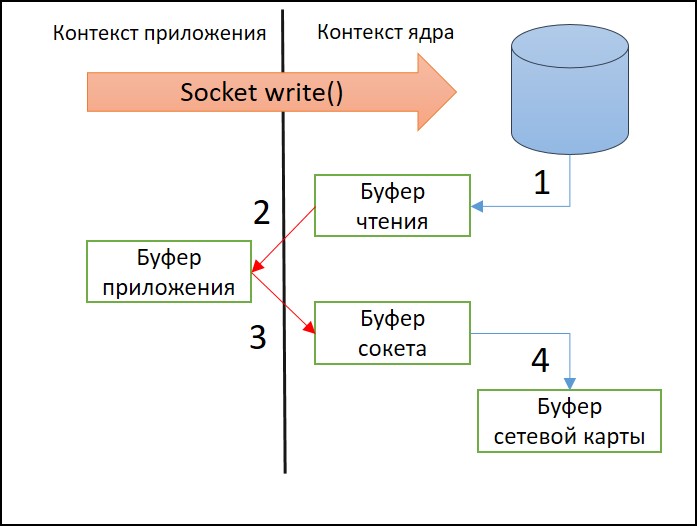


Рис. 3. Последовательность при передаче данных посредством Sockets интерфейса для TCP/IP стеке.

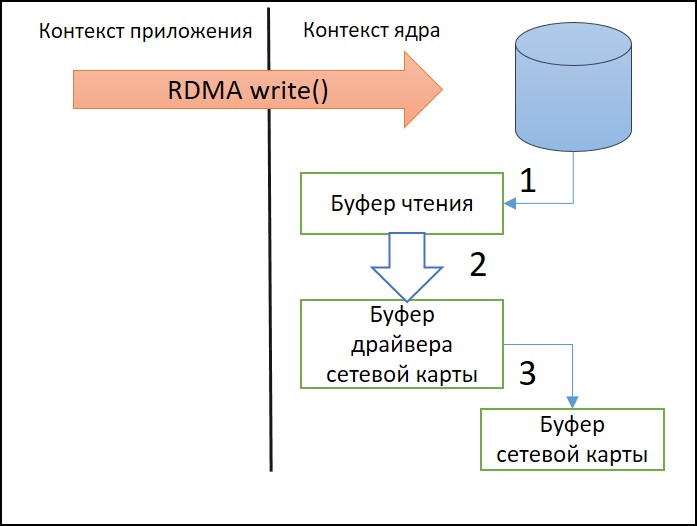


Рис.4. Последовательность при передаче данных посредством RDMA механизма для InfiniBand.

Заключение

Текущее состояние развития сетевых технологий позволяет использовать современные сетевые среды и оптимальные механизмы для передачи/приема данных, но появлением новой сетевой технологии влечет за собой создание нового программного интерфейса для разработчиков. Такая тенденция ведет к созданию проприетарных реализаций MPI протокола, которые являются непортируемыми на другие системы.

Создание унифицированного интерфейса, который позволит оптимально, с минимальными накладными расходами обеспечить портируемость на все современные сетевые технологии, является важной задачей, стоящей перед разработчиками и исследователями современных HPC сообществ.

За последние пару лет появились два конкурирующим между собой интерфейса - OFI (Open Fabric Interfaces)[2], разработанный Open Fabric Alliance, и UCX (Unified Communication X)[3], разработанный OpenUCX сообществом. Современные реализации MPI переходят на использование этих интерфейсов для обеспечения лучшей переносимости.

Литература

1. Tom Shanley, MindShare, Inc. InfiniBand Network Architecture, Addison-Wesley Professional, 2003;
2. OpenFabrics Interfaces Working Group - <https://www.openfabrics.org/>; <https://github.com/ofiwg>;
3. Unified Communication X Working Group - <https://github.com/openucx>; <http://www.openucx.org/>.