Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра суперкомпьютеров и квантовой информатики

Сорокина Надежда Вячеславовна

Использование технологии InfiniBand для передачи данных системы мониторинга суперкомпьютера

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

к. ф.-м. н.

Стефанов Константин Сергеевич

Москва, 2021

Оглавление

Оглавление 2

1. Введение 1

1.1 Цель работы 5

1.2 Задачи работы 5

1.3 Актуальность работы 6

2. Изучение принципов и особенностей функционирования системы DiMMon 6

3. Преимущество технологии InfiniBand с точки зрения реализации пользовательского приложения 9

4. Использование преимуществ технологии InfiniBand при разработке программных продуктов 11

5. Предоставляемые архитектурой InfiniBand типы взаимодействия пользовательских приложений 14

6. Программная реализация высокоуровневого интерфейса передачи данных между узлами суперкомпьютера поверх InfiniBand 15

6.1 Процесс установления канала сообщений между QP на узлах суперкомпьютера .. 16

6.2 Особенности разработки программного комплекса, реализующего передачу данных по сети InfiniBand с использованием транспортного режима UD 19

6.3 Реализация программного интерфейса для передачи данных между узлами суперкомпьютера 21

7. Тестирование и анализ исследования программного интерфейса передачи данных между узлами суперкомпьютера 25

8. Интеграция разработанного программного интерфейса в систему мониторинга суперкомпьютера DiMMon 26

9. Тестирование и анализ работы системы мониторинга суперкомпьютера DiMMon с внедренными модулями передачи данных по сети InfiniBand 31

10. Заключение 32

11. Дальнейшая работа 33

Список литературы 34

1. Введение

На сегодняшний день наблюдается значительный рост обрабатываемых данных. Высокопроизводительные вычисления необходимы практически в любой индустрии для решения огромного спектра задач от машинного обучения до моделирования реальных процессов. Вследствие существенного роста данных, которые требуется хранить и обрабатывать, возникает необходимость в огромных вычислительных мощностях. С точки зрения сетевых технологий данная тенденция говорит о том, что требуется более высокая пропускная способность канала передачи данных на всем пути, в том числе и между вычислительными узлами. Одновременно с интенсивным увеличением объема данных, также быстро возрастает и производительность программного обеспечения, то есть программные продукты становятся более требовательными к аппаратным платформам, в первую очередь к вычислительным серверам и к системам хранения данных. Для того, чтобы удовлетворять этим требованиям, производители оборудования увеличивают производительность своих систем. Эта тенденция показывает, что центры HPC и центры обработки данных нуждаются в скоростях, которые на сегодняшний день исчисляются уже сотнями гигабит в секунду.

Развитие компьютерных технологий и возросшая нагрузка на датацентры и корпоративные вычисления стимулируют улучшение существующих технологий соединения между вычислительными узлами. Концепции высокопроизводительных вычислений, такие как кластеризация, отказоустойчивость и круглосуточная доступность, требуют большей емкости для перемещения данных между узлами обработки. Одновременно с этим растут и требования к снижению задержки на уровне программных приложений. В рамках сформировавшихся требований многие известные технологии, которые используются на сегодняшний день, перестают удовлетворять современным запросам точной и быстрой доставки между устройствами. Применение технологии InfiniBand, обладающей высокой пропускной способностью и предоставляющей эффективное взаимодействие сетевых устройств, стало распространенным решением этой проблемы.

InfiniBand (сокращенно **IB**) — это высокоскоростная коммутируемая компьютерная сеть, используемая в высокопроизводительных вычислениях, а также для внутренних соединений в некоторых вычислительных комплексах. InfiniBand имеет пропускную способность до 200 гигабит в секунду [1] в зависимости от различных режимов и количества линий. Одними из основных принципов этой технологии являются

повышение масштабируемости, снижение загрузки центрального процессора и увеличение пропускной способности, позволяющие наращивать производительность и продолжать работу при выходе из строя части узлов. Архитектура InfiniBand возникла в 1999 году, и разрабатывалась как протокол межсистемных соединений между серверами или вычислительными узлами. В связи со спецификой рассматриваемой технологии, InfiniBand изначально реализована как высокоскоростная технология, которая позволяет передавать данные и через другие существующие протоколы с более высокой скоростью, а также использовать более надежную систему доставки сообщений. Отличительная особенность технологии заключается в оптимизированности для работы на уровне приложений, что дает возможность обеспечивать минимальные задержки в работе непосредственно приложения. Согласно списку самых мощных мировых вычислительных систем, составленному проектом Top500 в ноябре 2020 года [2], InfiniBand является популярной сетью для суперкомпьютеров в настоящее время, поскольку используется более чем в 30% систем.

Важной частью функционирования суперкомпьютеров является система мониторинга, обеспечивающая сбор данных о процессах, состоянии компонентов вычислительного комплекса и обрабатывающая полученные результаты для решения поставленных перед системой задач. В частности, мониторинг производительности суперкомпьютеров необходим для наблюдения за использованием вычислительных ресурсов системы и увеличения производительности. На суперкомпьютерах «Ломоносов» и «Ломоносов-2» Московского государственного университета имени М.В. Ломоносова в качестве системы мониторинга производительности реализована Динамически перестраиваемая Распределенная Модульная Система мониторинга суперкомпьютера (Dynamically Reconfigurable Distributed Modular Monitoring System for supercomputers, DiMMon) [3]. Эта система состоит из агентов, запускаемых на вычислительных узлах и передающих друг другу сообщения. Агенты состоят из модулей, имеющих некоторую функциональность с определенным при запуске агента интерфейсом. Для корректной работы системы агенты должны передавать друг другу данные по сети, чтобы связать функционал различных модулей. Из этого следует, что одним из наиболее значимых модулей является модуль передачи данных. В системе DiMMon этот модуль реализован с помощью технологии Ethernet и протокола UDP. Отличительной особенностью данной технологии является необходимость обращения приложения к ядру операционной системы для выделения сетевых ресурсов при передаче данных между удаленными приложениями.

В мировом опыте существует практика применения технологии InfiniBand. Система Lightweight Distributed Metric Service (LDMS), построенная с помощью этой технологии, используется для непрерывного мониторинга крупномасштабных вычислительных систем и приложений [4]. LDMS была разработана Национальным центром суперкомпьютерных приложений в университете Иллинойса. Использование технологии InfiniBand позволило уменьшить влияние системы мониторинга на выполнение вычислительных задачи. Также было получено представление о поведенческих характеристиках отдельных приложений в отношении использования ресурсов платформы (например, памяти, процессора, сети, питания) и о том, как эти ресурсы распределяются, нагружаются и истощаются из-за совокупной рабочей нагрузки.

Реализация передачи данных с использованием сети InfiniBand в системе DiMMon даст возможность обойти особенности технологии Ethernet и использовать ориентированность InfiniBand на сокращение задержки в работе приложений. Опыт американских разработчиков в построении системы мониторинга с транспортировкой данных по сети InfiniBand говорит о доступности применения описанных преимуществ.

1.1 Цель работы

Целью данной научной работы является реализация возможности использования технологии InfiniBand в качестве транспорта для передачи данных в системе мониторинга суперкомпьютера.

1.2 Задачи работы

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) изучить особенности работы с технологией InfiniBand и интерфейсами IB verbs и RDMA CM в процессе программирования;
2) разработать библиотеку, реализующую интерфейс для передачи данных по сети InfiniBand между узлами суперкомпьютера;
3) на основе разработанной библиотеки реализовать программный модуль DiMMon и интегрировать его в развернутую на суперкомпьютере «Ломоносов-2» систему мониторинга производительности DiMMon;
4) провести исследование производительности разработанной библиотеки и программного модуля на узлах суперкомпьютера «Ломоносов-2», а также в рамках системы мониторинга DiMMon.

1.3 Актуальность работы

Проведенная работа решает поставленную проблему – позволяет использовать все особенности и преимущества технологии InfiniBand при транспортировке данных в системе мониторинга DiMMon, и обойти принципы сетевой модели передачи данных TCP/IP, характерные для технологии Ethernet.

2. Изучение принципов и особенностей функционирования системы DiMMon

В настоящее время существует множество задач, решение которых без использования высокопроизводительных суперкомпьютерных систем и центров обработки данных не представляется возможным. Однако важно не только правильно решить поставленную задачу, но и правильно использовать вычислительные возможности суперкомпьютера. Очень часто наблюдается низкая эффективность работы, например, из-за неправильной программной реализации или некорректного использования вычислительных ресурсов. Важно следить за показателями поставленных на выполнение работ и выявлять причины неэффективной работы. Отслеживание подобных ситуаций является одной из задач системы мониторинга суперкомпьютера. Помимо отслеживания выполнения пользовательских приложений, также необходимо осуществлять мониторинг подсистем в самих суперкомпьютерах. Эффективная и согласованная работа всех составляющих суперкомпьютерной системы требует постоянного контроля и оперативного обнаружения отказов и ошибок, что может предоставить возможность проактивно выявлять потенциальные сбои и выходы оборудования из строя.

Система DiMMon решает поставленные задачи благодаря своей структуре, которая требует изучения особенностей ее работы для внедрения передачи данных по сети InfiniBand (рис 1.). Основными компонентами рассматриваемой системы являются *агенты мониторинга* – обычные процессы операционной системы, которые поддерживают работу узлов мониторинга и соединения между ними путем обмена сообщениями по сети. Агенты могут быть запущены на вычислительных узлах или других компьютерах.

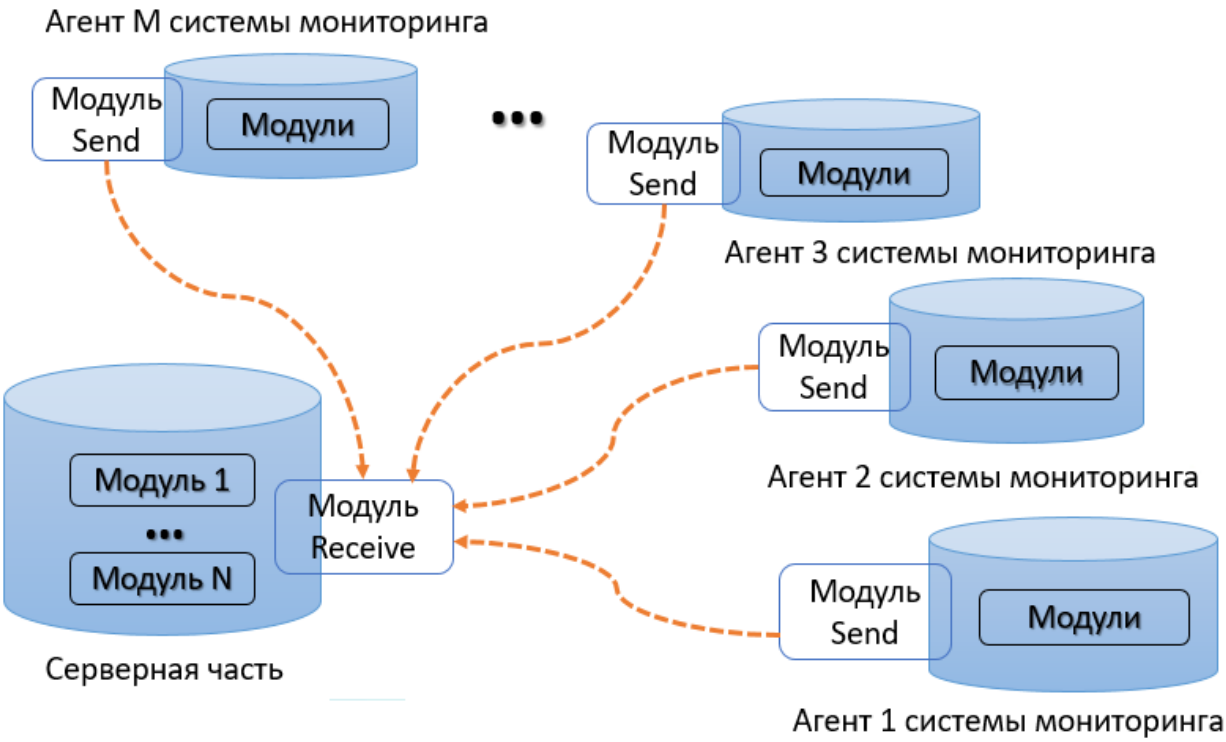

Рисунок 1. *Упрощенная схема работы системы мониторинга DiMMon.*

При создании узла мониторинга задается определенный тип и определяются функции, которые узел выполняет под управлением системы DiMMon [4]. Один или несколько типов узлов описываются в *модуле* - динамически подключаемой библиотеке с определенным интерфейсом. Взаимосвязь различных модулей устанавливается заданной конфигурацией при запуске агента мониторинга. Соединение между узлами создается путем указания имени или идентификатора исходного узла, имени выхода на текущем узле для установления соединения, получателя данных и имени входа, на который будут отправляться данные.

Необходимым компонентом для работы системы мониторинга являются таймеры. Таймер необходим для запуска определенных действий в определенные моменты времени. Таймер может срабатывать однократно или через определенные интервалы времени.

Для построения системы мониторинга используются несколько типов узлов. Однако единственное различие между этими типами заключается в их функциональности, поскольку сам агент мониторинга обрабатывает все узлы одинаково. Существуют следующие узлы:

- Узел, который получает данные мониторинга от аппаратного или программного обеспечения суперкомпьютера (оборудования или операционной системы) - датчик. Датчик - это узел, который не имеет входов. Его задача - дождаться сигнала

таймера, затем получить данные мониторинга, запросив аппаратное обеспечение, службы операционной системы и т.д., и отправить их на свой выход (датчики обычно имеют только один выход).

- Узел с входами и выходами обрабатывает данные мониторинга. Обработка может принимать различные формы: фильтрация данных, вычисление скорости, с которой изменяется значение.
- Узел, который имеет вход, но не имеет выходы, предназначен для отправки данных за пределы агента. Этот узел, скорее всего, будет использоваться для организации взаимодействия между агентами. Он получает данные мониторинга на свои входы, возможно как-то их преобразует и отправляет по сети другому агенту, работающему на том же или другом компьютере. Принимающий агент управляет принимающим узлом, который не имеет каких-либо входов, как сенсорный узел. Но в отличие от датчика, он получает данные не от операционной системы или оборудования, а от другого агента по сети.

При запуске агента мониторинга происходит чтение файла конфигурации, который содержит список загружаемых модулей, а после - их загрузка. Набор типов узлов, которые задают функциональность агента мониторинга, определяется типами, описанными в загруженных модулях.

В рамках рассмотренных принципов функционирования системы мониторинга DiMMon модуль передачи данных имеет два типа узлов – узлы, инициирующие передачу данных, и принимающие узлы. Этот модуль реализован с помощью технологии Ethernet и сокетов UDP из стека протоколов TCP/IP. Система DiMMon просматривает имеющиеся открытые файловые дескрипторы в ожидании данных, которые там должны появиться, то есть рассматриваемая система построена с использованием технологии epoll. В рамках поставленных в работе задач необходимо разработать подобный функционал, который не нарушает логику работы рассматриваемой системы. Для организации передачи данных между принимающим и отправляющим узлом необходимо реализовать и протестировать отдельный программный интерфейс, который позволит упростить использование технологии InfiniBand в системе DiMMon. Этот программный интерфейс должен использовать клиент-серверную технологию, а также предоставлять гибкую настройку параметров работы модуля.

3. Преимущество технологии InfiniBand с точки зрения реализации пользовательского приложения

Традиционно в высокопроизводительных системах, где производится обработка и хранение больших объемов данных, а также их транспортировка между вычислительными узлами, всеми процессами, выполняющими работу с данными, управляет операционная система и делает доступными ресурсы для пользовательских приложений по мере необходимости. Это говорит о том, что приложение, не имея прямого доступа к сети, должно с помощью различных API выполнять запрос к операционной системе, которая выделяет сетевые ресурсы для передачи данных из буферного пространства приложения (рис.2). Аналогичная ситуация происходит и при получении данных на принимающей стороне. Такой принцип построения передачи данных свойственен и для технологии Ethernet, которая используется в системе DiMMon. Однако существует и другой подход к организации транспортировки данных между программами.

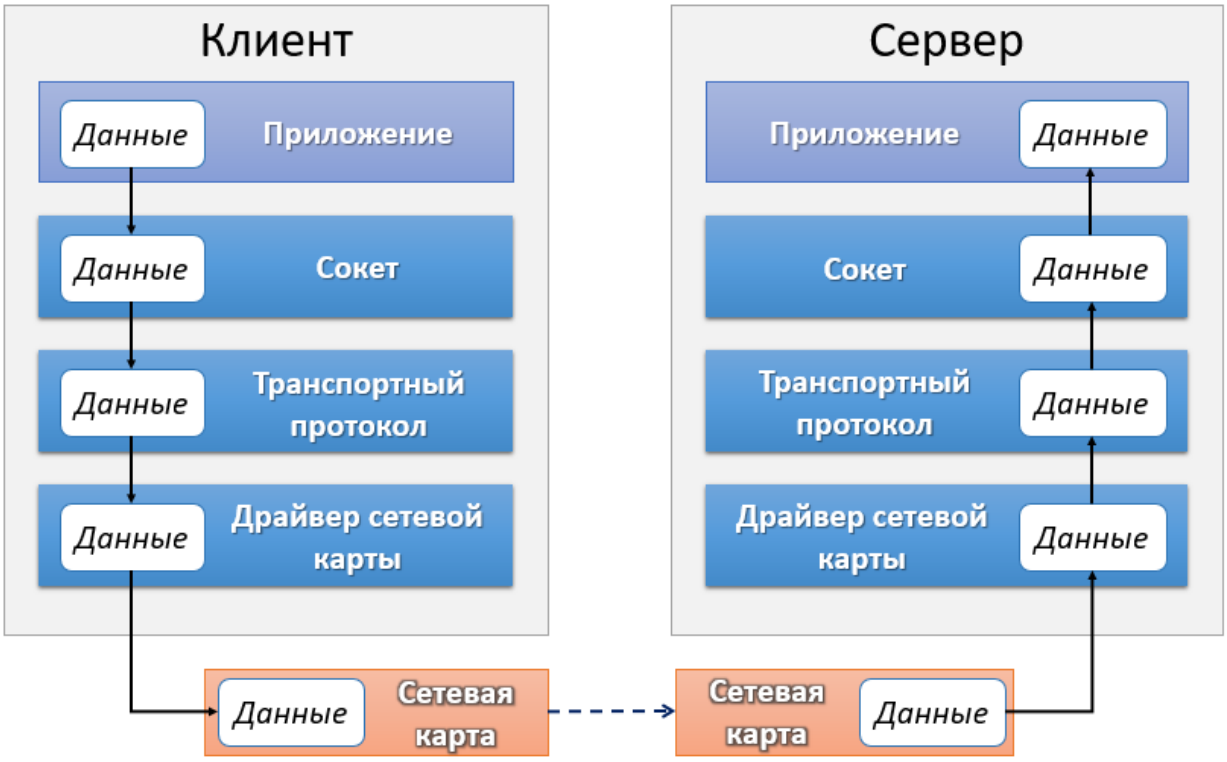

Рисунок 2. *Принцип организации передачи данных с использованием технологии Ethernet, основанный на обращении приложения к операционной системе для выделения сетевых ресурсов.*

Разработка архитектуры InfiniBand как технологии, предоставляющей минимальную задержку на уровне программных приложений, имеет важную характерную особенность [5]. Она заключается в том, что приложение может напрямую обращаться к

сетевому интерфейсу при необходимости передачи данных другому приложению в обход центрального процессора, то есть предоставляет возможность обмена сообщениями напрямую между соответствующими виртуальными буферами по сети (рис.3). При этом операционная система не участвует в таком сетевом взаимодействии. Технология позволяет создать изолированный защищенный «виртуальный канал», напрямую соединяющий два приложения, которые могут находиться на разных вычислительных узлах. Этот канал может использоваться для связи с другими приложениями или процессами или для доступа к хранилищу. Вместо того, чтобы делать запрос к операционной системе для доступа к одному из коммуникационных ресурсов сервера, приложение напрямую обращается к созданному каналу обмена сообщениями InfiniBand. Это могут быть как пользовательские приложения, так и приложения ядра, как, например, файловая система. Поскольку трафик сети можно рассматривать как совокупность управляющих сообщений и сообщений с данными, одна и та же служба обмена сообщениями одинаково удобна для использования и в приложениях, и для различных управляющих систем, в том числе и для системы мониторинга суперкомпьютера.

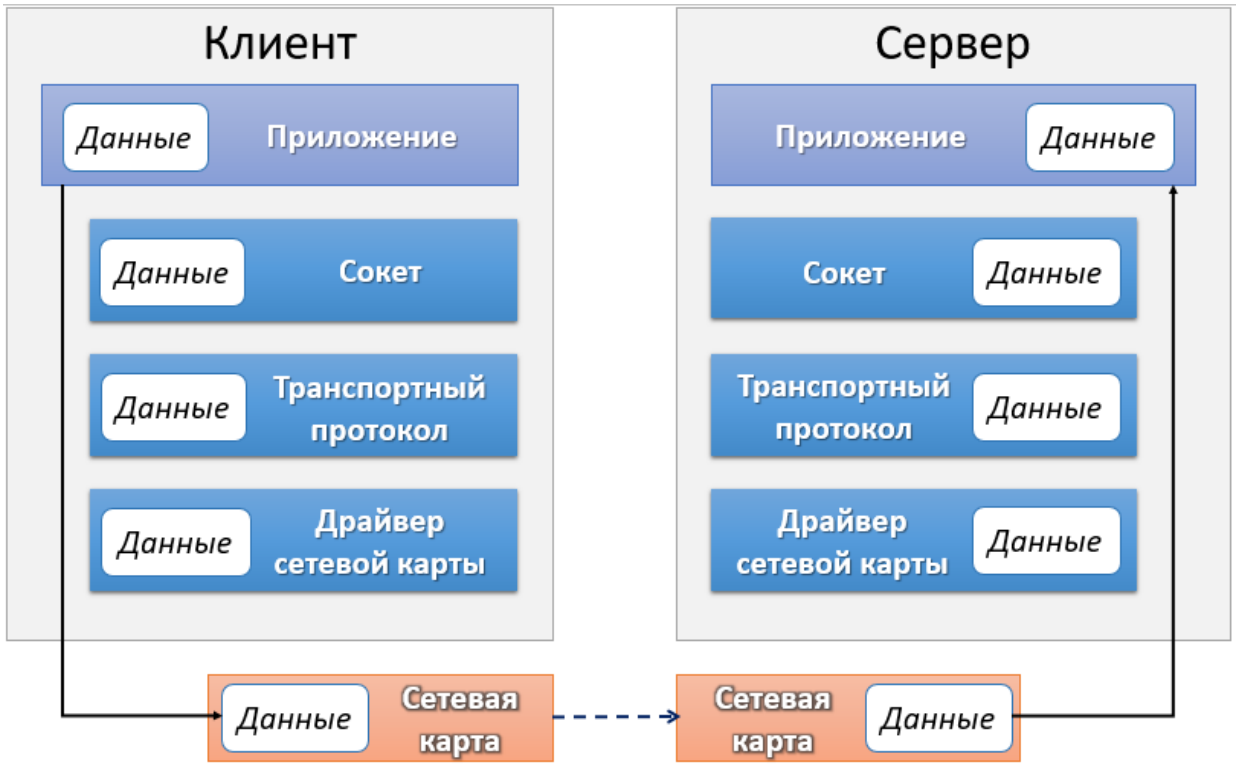

Рисунок 3. *Концепция прямого доступа из приложения непосредственно к сетевому интерфейсу в обход центрального процессора, предоставляемая технологией InfiniBand.*

Такая концепция InfiniBand отличается от принципов построения взаимодействия в традиционной сети Ethernet, хоть и имеет схожую со стеком протоколов TCP/IP сетевую

концепцию. Основная особенность InfiniBand заключается в создании канала обмена сообщениями, в который приложения могут отправлять запросы на пересылку данных, что является ключевым отличием от технологии Ethernet, который представляет собой транспорт для передачи байтов информации между сокетами.

Вся архитектура InfiniBand построена вокруг идеи возможности прямого взаимодействия приложений путем обмена сообщениями через канал. Поскольку такая реализация значительно отличается от принципов традиционных технологий, использование канала обмена сообщениями в программных модулях имеет свои трудности и особенности. Для разработки приложений, передающих данные с помощью технологии InfiniBand, необходимо изучить принципы программирования и правильного применения всех компонентов, поддерживающих работу канала обмена сообщений InfiniBand.

4. Использование преимуществ технологии InfiniBand при разработке программных продуктов

Рассмотренные в предыдущем пункте особенности архитектуры InfiniBand вносят значительные изменения в логику построения пользовательского приложения. Поэтому данный раздел будет посвящен основам программирования приложений, использующих InfiniBand в качестве транспорта для передачи данных. Для этого необходимо рассмотреть принцип работы непосредственно с изолированным каналом обмена сообщениями между взаимодействующими приложениями.

Основной механизм, который лежит в основе коммуникаций между конечными точками, называется пара очередей (Queue Pair, сокращенно QP) [6]. Это примерный эквивалент сокета в стандартном IP сети. Каждая пара очередей состоит из очереди отправки и очереди приема на каждом конце созданного канала. Пара очередей - это обычная структура с точки зрения программирования, с помощью которой приложение обращается к каналу обмена сообщениями InfiniBand. Эта структура связывает данные приложения с данными приложения на другом конце защищенного канала, предоставляющего механизмы для передачи сообщений непосредственно между приложениями.

Следующий механизм, необходимый после создания канала – это непосредственно передача сообщения. InfiniBand предоставляет два варианта – передача данных с использованием канала, называемая Send/Receive, и передача данных путем непосредственного взаимодействия с памятью удаленной стороны, называемая RDMA

Read и RDMA Write. В первом случае отправленные данные принимаются в виде структуры, переданной в канал сообщений, откуда технология InfiniBand переносит ее в очередь приема удаленной стороны. Таким образом, передающая сторона не знает о созданных на принимающей стороны структурах данных и устройстве памяти; вместо этого она просто выполняет операцию отправки сообщения – Send, а принимающая – операцию получения Receive. Во втором случае реализован несколько иной принцип. Принимающая сторона выделяет область памяти необходимого размера, регистрирует ее и передает управление удаленной стороне. В свою очередь удаленная сторона для взаимодействия использует операции RDMA Read или RDMA Write для чтения или записи данных из этой области памяти соответственно. Принцип передачи данных по каналу обмена сообщений основан на постановке запросов к паре очередей на выполнение отправки или приема данных – Send Work Request (Send WR) и Receive Work Request (Receive WR) соответственно. Такие запросы WR представляют собой единичный объем работы, который приложение может выполнить с каналом обмена сообщениями.

Для организации передачи данных в системе мониторинга наилучшим вариантом является первый механизм, поскольку второй имеет явный недостаток в поставленных условиях. При использовании RDMA операций сторона, отдавшая управление своей памятью, не информируется о завершении произведенных операций. Для обнаружения изменения данных записывающей стороной в памяти выставляется флаг, который принимающая сторона должна проверять в цикле. Циклический опрос потребляет процессорное время, кроме того возрастают накладные расходы памяти и задержки для систем с очень большим количеством узлов. Однако в случае использования операций Send/Receive инициирующая передачу данных сторона может выполнять другие операции, в то время как удаленная точка асинхронно завершила операцию Receive принимающей стороны. Это возможно благодаря очереди завершения запросов – Completion Queue (CQ), что является важным преимуществом InfiniBand в рамках системы мониторинга суперкомпьютера, благодаря которому не происходит неэффективного использования вычислительных ресурсов.

InfiniBand снимает нагрузку управления трафиком с программного клиента за счет использования очереди завершения [7]. Каждая сторона создает такую очередь, ассоциирует ее с уже аллоцированной QP, а затем передает ее InfiniBand для управления. Клиент помещает запрос в указанную пару очередей (при этом такой WR становится WQE – Work Queue Entry). Когда выполнение поставленной единичной работы завершается, информация об успешности выполнения WQE возвращается в CQ в виде схожей по семантике единичной работе канала завершений – Completion Queue Entry (CQE).

Технология InfiniBand позволяет пользователю канала обмена сообщениями ставить сразу несколько запросов на передачу данных WQE, что дает возможность выполнять другую работу во время обработки WQ. Рисунок 4 схематично показывает взаимосвязь между QP, WQE и CQE.

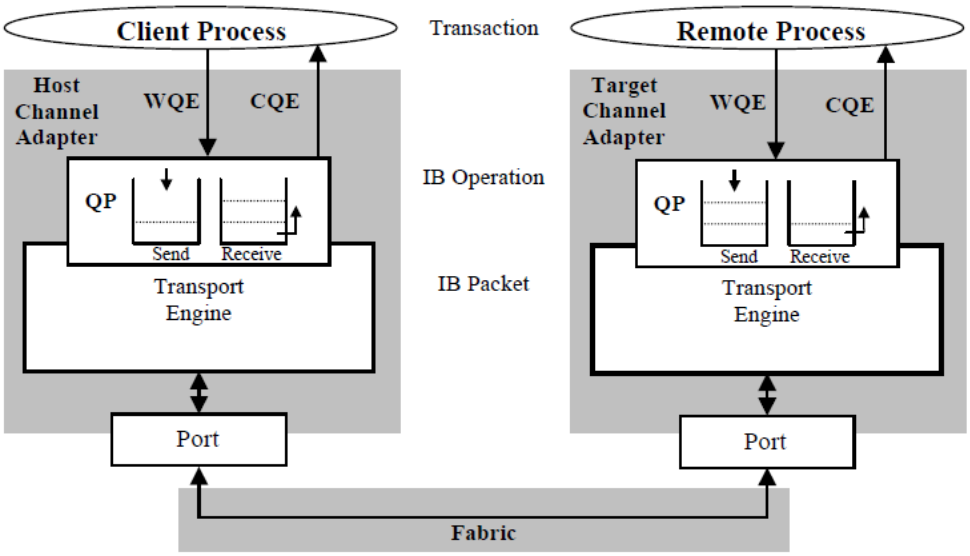

Рисунок 4. *Модель коммуникаций приложений с помощью системы транспорта сообщений InfiniBand (рис.10 из [7]).*

Для организации передачи данных транспортный интерфейс InfiniBand определяет набор определенных методов, которые приложение использует для запроса выполнения транспортировки данных к каналу обмена сообщениями. Например, для постановки запроса на отправку сообщения используется функция ibv_post_send, которая отправляет список рабочих запросов WR в очередь отправки пары очередей QP.

Стоит отметить, что спецификация архитектуры InfiniBand не определяет реальных API; это остается на усмотрение других организаций, таких как OpenFabrics Alliance. Эта организация предоставляет полный набор API-интерфейсов с открытым исходным кодом и программным обеспечением, которое реализует методы и компоненты, функционирующие на оборудовании InfiniBand. Таковой является библиотека IB verbs (libibverbs) — низкоуровневый универсальный программно-аппаратный интерфейс (API), с помощью которого производилась разработка программного интерфейса.

5. Предоставляемые архитектурой InfiniBand типы взаимодействия пользовательских приложений

В спецификации InfiniBand указано четыре различных типа передачи данных между приложениями: надежное соединение (Reliable Connection, RC), ненадежное соединение (Unreliable Connection, UC), надежная датаграмма (Reliable Datagram, RD) и ненадежная датаграмма (Unreliable Datagram, UD) [8]. Эти транспортные режимы можно настроить при установке QP. Также от них зависят многие другие параметры ibverbs компонентов, в том числе и операции передачи и получения данных. Стоит отметить, что RD не поддерживается этим API. Разница между надежным и ненадежным аналогична разнице между TCP и UDP - при надежном соединении данные передаются по порядку, и их доставка гарантируется. В случае ненадежного соединения нет никаких гарантий доставки сообщений и их очередности. Тип соединения - это ассоциация строго между двумя хостами. В дейтаграмме хост может свободно взаимодействовать с любым другим хостом в подсети. Рассмотрим подробнее предлагаемые типы соединений:

- Надежное соединение (RC). Пара очередей одной стороны соединения связана только с одним QP другой стороны. Сообщения, передаваемые с помощью этого режима, доставляются надежно, пакеты доставляются по порядку. RC-соединение очень похоже на TCP-соединение в стеке TCP/IP.
- Ненадежное соединение (UC). Пара очередей одной стороны соединения связана только с одним QP другой стороны, аналогично RC. Однако это соединение является ненадежным, поэтому пакеты могут быть потеряны. Сообщения о потере пакетов или сообщения с ошибками не посылаются повторно, поэтому обработка ошибок должна обеспечиваться протоколом более высокого уровня.
- Ненадежная дейтаграмма (UD). Пара очередей может передавать и принимать сообщения в/от любого QP, настроенного на UD передачу, противоположной стороны. Доставка пакетов не гарантируется, кроме того уже доставленные пакеты могут быть отброшены получателем. Поддерживаются многоадресные сообщения (от одного ко многим) – Multicast-рассылка. Соединение UD очень похоже на соединение UDP стека TCP/IP. Важно отметить, что в отличие от других типов передачи данных, которые имеют максимальный размер сообщения 1 Гбайт, UD имеет ограничение размером Maximum Transport Unit (MTU). Этот параметр зависит от InfiniBand устройства, и может быть перенастроен. Информация о нем содержится в специальной структуре ibv_mtu в настройках QP.

Описание предоставляемых InfiniBand транспортных режимов необходимо учитывать при разработке пользовательских приложений. Согласно поставленной цели работы самым подходящим вариантом для разработки программного комплекса, использующего передачу данных по InfiniBand, является UD. Надежное и ненадежное соединение требуют наличия информации о каждом соединении (как минимум, информация об удаленной QP), которое создается с каждой противоположной стороной. Хотя UD не гарантирует доставку сообщений, его модель без установления соединения сокращает ресурсы, необходимые для связи, и обеспечивает повышение производительности при взаимодействии с множеством различных соединений. Поскольку вычислительных узлов в суперкомпьютерных системах большое количество, накладные расходы на хранение информации о соединениях могут привести к неприемлемо неэффективному использованию памяти и увеличению времени установления соединения. Это не оправдывается предоставляемым, например, RC преимуществом – надежностью доставки данных, которого лишен UD. Однако этот недостаток UD в системе мониторинга может быть решен интерполяцией данных при потере пакета. Рассмотренные характеристики UD делают его удобным и наиболее правильным вариантом в качестве транспортного протокола для поддержки InfiniBand в системе мониторинга суперкомпьютера.

6. Программная реализация высокоуровневого интерфейса передачи данных между узлами суперкомпьютера поверх InfiniBand

Решение одной из поставленных в работе задач заключается в разработке высокоуровневого интерфейса прикладного программирования (API), позволяющего осуществлять передачу данных между узлами суперкомпьютера по сети InfiniBand с использованием интерфейса IB verbs. Высокоуровневость программной реализации дает преимущество его применения – пользователю не нужно быть профессионалом в этой узкоспециализированной области, не нужно знать технических тонкостей работы с InfiniBand. Для написания приложения, передающего данные по сети InfiniBand, пользователь вызывает лишь функции, которые реализуют, например, обращение к серверу. Разработанный программный интерфейс позволяет создать клиентскую и серверную часть (приложение), работающую в режиме множественного доступа, которые используют передачу данных с помощью UD.

Задачу написания такого высокоуровневого интерфейса необходимо разбить на несколько подзадач:

- Изучение процесса установления соответствия между QP участвующих в коммуникации сторон;
- Изучение особенностей транспортного режима Unreliable Datagram при использовании интерфейса IB verbs;
- Непосредственно реализация на основе изученных принципов программного интерфейса, предоставляющего удобную функциональность для передачи данных между узлами суперкомпьютера.

6.1 Процесс установления канала сообщений между QP на узлах суперкомпьютера

Специально для установления канала сообщений между сторонами, участвующими в коммуникации поверх сети InfiniBand, был разработан интерфейс прикладного программирования RDMA CM (RDMA Communication Manager) организацией Open Fabrics Alliance. RDMA CM – это диспетчер связи, используемый для настройки транспортного режима – RC, UC или UD. Он предоставляет промежуточный транспортный интерфейс, концепции которого основаны на сокетах, но адаптированы для семантики на основе пар очередей (QP): связь должна осуществляться через устройство, поддерживающее технологию InfiniBand; транспорт данных осуществляется на основе передачи сообщений, а не на основе потока байт.

RDMA CM осуществляет управление коммуникацией, устанавливая или разрывая соединение, однако помимо этого в него заложена возможность работы и с компонентами IB verbs в виде более высокоуровневых функций. Этот интерфейс работает вместе с компонентами, определенными библиотекой libibverbs. Важной особенностью RDMA CM является возможность работы как синхронно, так и асинхронно – такой режим работы необходим для эффективной функциональности системы мониторинга.

В начале коммуникации сторонам необходимо настроить локальную QP так, чтобы они указывали на соответствующую QP на удаленном узле. Однако в QP нет заложенной возможности найти друг друга по сети. Так как же происходит установление соединения между узлами суперкомпьютера по сети InfiniBand? Вся требуемая для ассоциации QP информация передается с помощью RDMA CM по аналогии с сетевым сокетом. Основными параметрами, которые необходимы для InfiniBand, и которые используются интерфейсом, являются компоненты Queue Pair Number(QPN), Local Identifier (LID) и Packet Sequence Number (PSN) [9]. Эта тройка параметров определяет «адрес» QP удаленного узла (рис.5).

Остановимся на них более подробно:

- Queue Pair Number - это номер QP - идентификатор, назначенный каждой очереди в HCA (Host Channel Adapter) устройства InfiniBand, который будет использоваться для указания того, в какую очередь следует отправлять сообщения (и из какой считывать);
- Local Identifier - это уникальный номер, который присваивается каждому порту устройства, когда он становится активным;
- Packet Sequence Number - это порядковый номер пакета. В надежном соединении PSN используется, чтобы проверить, что пакеты идут в правильном порядке и что ни один пакет не пропущен. Если PSN совпадает с уже использованным PSN, аппаратное обеспечение будет считать, что полученный пакет является устаревшим пакетом из старого соединения, и сбросит его.

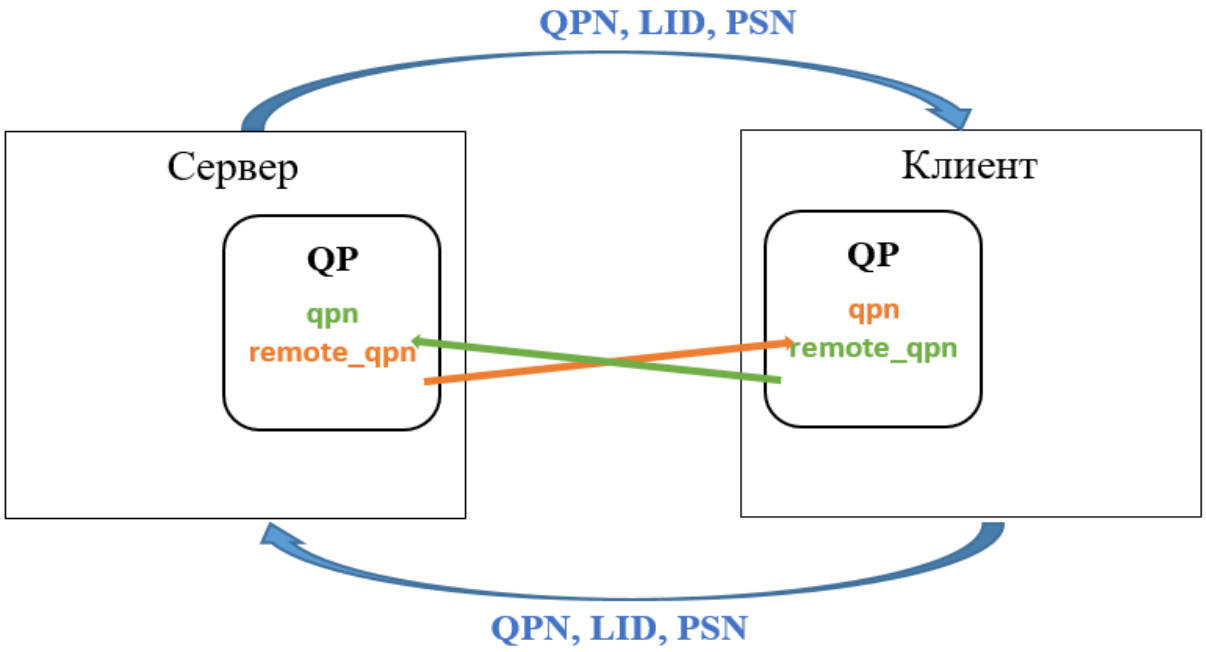

Рисунок 5. *Упрощенная модель установления канала сообщений между парой очередей участвующих в передаче данных поверх InfiniBand узлов суперкомпьютера.*

Сам процесс установления канала сообщений между QP контролируется пользователем с помощью канала событий (Event Channel) в определенных вызовах функций RDMA CM. Создание такого канала событий предоставляет возможность идентификатору соединения (rdma_cm_id) сообщать данные о событиях, произошедших в этом канале. Event Channel информирует стороны, участвующие в установлении соединения, о намерениях и подтверждениях противоположной стороны, а также об инициировании собственных событий. Например, такими событиями могут быть запрос

клиента к серверу на подключение, ответ сервера на такой запрос, запрос на прекращение передачи данных и отсоединение, и другие. Если канал событий создан не будет, то все операции RDMA CM для выбранного идентификатора будут заблокированы до их завершения. Из полученных событий канала извлекается необходимая информация о противоположной стороне. Например, серверная часть получает структуру ibv_context, которая используется для создания таких компонент, как CQ, QP и других. В свою очередь клиентская часть получает требуемую для транспортного режима UD структуру ibv_ah_attr для настройки еще одного важного для передачи параметра Address Handler (AH). Эта компонента IB verbs будет рассмотрена в следующем пункте.

Для того, чтобы правильно настроить канал сообщений, в серверной части необходимо создать все требуемые компоненты, выделить порт, который будет в состоянии Listen – в состоянии ожидания подключений от клиентов, пока не произойдет событие RDMA_CM_EVENT_CONNECT_REQUEST, обозначающее желание клиента создать канал сообщений к серверу. Для работы с каналом событий применяется блокирующая функция get_cm_event, которая снимает блокировку, если оповещение пришло. Однако блокировка означает, что вычислительные мощности сервера не будут использоваться эффективно – вместо решения поставленных клиентами задач или вместо ответа другим клиентам, сервер будет простаивать, что губительно скажется на системе мониторинга. Решение этой проблемы также будет рассмотрено в следующем пунке. Установив канал с клиентской частью функцией accept, сервер готов к получению данных.

Программа клиента имеет значительное отличие. В начале клиентский модуль аналогично создает все необходимые компоненты, затем проверяет доступность сервера через переданный в качестве параметров при запуске адрес – IP-адрес и порт удаленного узла, на котором запущен сервер. Следующим шагом он вызывает функцию connect, после чего в блокирующем режиме ожидает события RDMA_CM_EVENT_ESTABLISHED – оповещения о том, что сервер готов принимать данные. В этот момент начинается передача информации.

На данном этапе важно отметить значимую деталь, поскольку может возникнуть проблема отсутствия данных при условии, что подключение произошло успешно, а удаленная сторона правильно организовала отправку. Причиной может быть несвоевременное выделение сервером буфера для получения данных - после принятия запроса на соединение. Обосновано это тем, что клиент может отправить данные быстрее, чем сервер будет готов их получить, поэтому функция ibv_post_recv, подготавливающая

место для новых данных, всегда должна вызываться раньше, чем отправляющая сторона поставит данные на отправку функцией ibv_post_send.

6.2 Особенности разработки программного комплекса, реализующего передачу данных по сети InfiniBand с использованием транспортного режима UD

Использование Unreliable Datagram в качестве транспортного режима при разработке программ требует рассмотрения некоторых принципов, которые возникают благодаря характерным чертам UD – отсутствие установления сетевого соединения и ненадежность доставки данных. При программировании стоит учитывать следующие аспекты:

1) *Настройка пары очередей.* В первую очередь нужно правильно задать характеристику создаваемых QP. Для этого в структуре qp_attr параметр типа задается как IBV_QPT_UD. Эта структура в дальнейшем используется при создании самой QP;

2) *Порядок событий в процессе создания канала сообщений между QP*. Порядок получения событий в канале событий при установлении соединения между QP, предоставляемого механизмом RDMA CM, отличается для различных транспортных режимов. Для UD событие типа RDMA_CM_EVENT_ESTABLISHED – событие установления соединения - не происходят в серверной части, поскольку UD подразумевает передачу датаграммы, а не сетевое соединение для передачи данных (аналогично UDP). По этой же причине не происходит и события RDMA_CM_EVENT_DISCONNECTED – события разрыва соединения для обеих сторон общения.

3) *Настройка параметров передаваемых данных*. Данные в технологии InfiniBand транспортируются в виде структуры ibv_send_wr. Указанная структура инициирует посылку данных, поэтому ее необходимо дополнить требуемыми компонентами, которые будут использованы при передаче сформированной датаграммы. Этими компонентами являются:

   - Структура Address Handle (AH) – это объект, описывающий путь к удаленной стороне. Поля данной структуры содержат Local Identifier (LID), Global Identifier (GID) и другую информацию, необходимую для маршрутизации сообщения по сети InfiniBand. В транспортных режимах, устанавливающих

соединение (RC, UC) AH связан с парой очередей (QP). В режимах передачи дейтаграмм (UD) AH связан с Work Request (WR).

- Значение Remote Queue Pair Number (QPN) – номер пары очередей QP узла-адресата;
- Значение Remote Queue Key (QKey) – ключ, который генерируется и ассоциируется с парой очередей QP удаленным узлом. Он требуется для ограничения доступа к QP, чтобы избежать преднамеренного или случайного обращения к ней без наличия разрешения от принимающей стороны.

Наличие этих полей обязательно для того, чтобы сформированная датаграмма была доставлена и имела доступ к паре очередей QP, ассоциированной с установленным соединением на противоположной стороне.

4) *Использование при транспортировке сообщения дополнительного заголовка*. Это влечет за собой изменение размера выделяемой под получаемые данные памяти на принимающей стороне. Такой заголовок называется заголовок глобальной маршрутизации - Global Routing Header (GRH) [10] - и используется для реализации возможности транспортировки сообщения через границу подсети в InfiniBand, а также реализации для многоадресной маршрутизации по сети (Multicast). Добавление GRH области к телу сообщения для хранения информации о глобальной маршрутизации, благодаря чему противоположная сторона сможет сгенерировать соответствующий вектор адресов для ответа на принятый пакет. Обычно данный заголовок занимает 40 байт, поэтому при чтении полученных данных из выделенной для них памяти необходимо учитывать указанный сдвиг.
5) *Генерация сообщения отвечающей стороной*. Как уже говорилось во втором и третьем пункте, UD не устанавливает сетевое соединение, а значит и не получает события RDMA_CM_EVENT_ESTABLISHED, благодаря которому транспортные режимы с установлением соединения всегда знают QPN, QKey и AH, с которыми сразу могут ассоциировать свои компоненты QP. В случае UD требующуюся для этого информацию необходимо извлекать из WC, сгенерированного после получения данных в CQ. Использование функции ibv_create_ah_from_wc, а также присваивание QKey и QPN соответствующим полям структуры ibv_send_wr при посылке данных гарантирует корректную передачу сообщения до удаленной стороны. При наличии гарантий, что общение идет с одной и той же QP противоположный стороны, описанный процесс можно выполнить только при

первом ответе, далее вычисленные компоненты не изменятся. В ином случае необходимо проделывать эти операции при каждом полученном запросе.

6.3 Реализация программного интерфейса для передачи данных между узлами суперкомпьютера

Изучение основных принципов установления канала сообщений между QP и транспортировки данных с помощью Unreliable Datagram позволяет перейти к разработке высокоуровневого интерфейса прикладного программирования, позволяющего осуществлять передачу данных между узлами суперкомпьютера по сети InfiniBand.

Традиционное программирование с помощью сокетов обычно использует технологию мультиплексирования IO (select, poll, epoll и т.д.) для создания механизма уведомления о событиях. Поскольку дальнейшим интересом для разработки является внедрение программного интерфейса в систему мониторинга DiMMon, необходимо учитывать принципы рассматриваемой системы при реализации серверной части. Поэтому механизм epoll лег в основу реализации возможности работы сервера с несколькими клиентами одновременно [11]. Эта технология дает ряд преимуществ для разрабатываемой программы:

- Узел, выступающий в качестве сервера, может поддерживать большое количество UD-каналов, т.е. обрабатывать запросы от многих клиентов;
- Узлы, которые долго не посылают данные, не тратят ресурсы и время работы сервера, поскольку он не блокируется на ожидании информации только от одного клиента;
- Наличие дополнительной возможности для серверной части запоминать каждый новый запрос на подключение, что позволяет работать с каждым клиентом отдельно;
- Разграничение каждого соединения также гарантирует отсутствие проблемы с синхронизацией потоков и доступом к общей памяти.

Для реализации данной технологии в разрабатываемом программном интерфейсе необходимо перенастроить все использующиеся каналы событий в неблокирующий режим. Речь идет о двух каналах событий – rdma_event_channel библиотеки RDMA CM, содержащем события соединения, и ibv_comp_channel библиотеки IB verbs, который хранит события об успешности завершения Work Requests – Send Request или Receive Request. Оба эти канала являются структурами, содержащими файловый дескриптор.

Именно его и необходимо сделать неблокирующим выставив соответствующий флаг O_NONBLOCK [12]. Для этого используется функция fcntl из библиотеки <fcntl.h>:

```
int flags = fcntl(comp_chan->fd, F_GETFL);
if (fcntl(comp_chan->fd, F_SETFL, flags | O_NONBLOCK) < 0)
        printf("Failed to change file descriptor of completion event channel\n");
```

Это позволяет организовать коммуникацию с многими клиентами через последовательный опрос Completion Channel каждого клиента, и при этом проверять Event Channel на новые уведомления о соединениях.

Требование удобной функциональности разрабатываемого программного интерфейса разрешается в пользу реализации программы на языке C++, что предоставляет весомое преимущество благодаря механизму наследования классов и перегрузке функций. Это позволяет произвести редукцию реализации принимающей (серверной) части на две подзадачи:

1) Реализация добавления и отключения клиентов в список коммуникаций, получение запросов и рассылка ответов (если требуется) для многих клиентов, а также проверка подключений от новых клиентов;
2) Реализация функциональности сервера – способа обработки принятых данных в случае реализации отвечающего сервера.

Реализация специальных классов в клиент-серверном комплексе, использующем передачу данных с помощью UD решает первую подзадачу (рис.6). Такими классами являются:

- Class ClientInfo - новый экземпляр этого класса создается сервером каждый раз, когда подключается новый клиент (и соответственно корректно удаляется после окончания сессии вместе со всеми выделенными компонентами). Он содержит необходимые для общения сервера с клиентом структуры данных, ассоциированные с этим соединением. Класс ClientInfo является вспомогательным и не доступен вне библиотеки, создавая отношение ассоциации (композиции) с классом Server.
- Abstract Class Server - этот класс содержит в себе высокоуровневые функции отправки и получения данных, добавления новых клиентов, запуска цикла обработки запросов от клиентов из массива экземпляров ClientInfo. Данный массив

является закрытым, поэтому никто не сможет инициировать неразрешенное соединение. Основные функции класса – CheckEventChannel и ProcessResponse, которые запускают процесс проверки наличия новых запросов коммуникации или пришедших данных соответственно. Они возвращают значение, равное -1, в случае отсутствия запросов на подключение и новых данных, и отрабатывают по очереди пока полностью не выполнят поставленную работу. Класс Server является абстрактным, поскольку содержит виртуальную функцию Factory – именно эта функция определяет функциональность сервера. Соответственно нельзя создать экземпляр этого класса.

- Class Client – класс, инициирующий обращение к серверу и отправляющий запросы с помощью функции GetRequest.

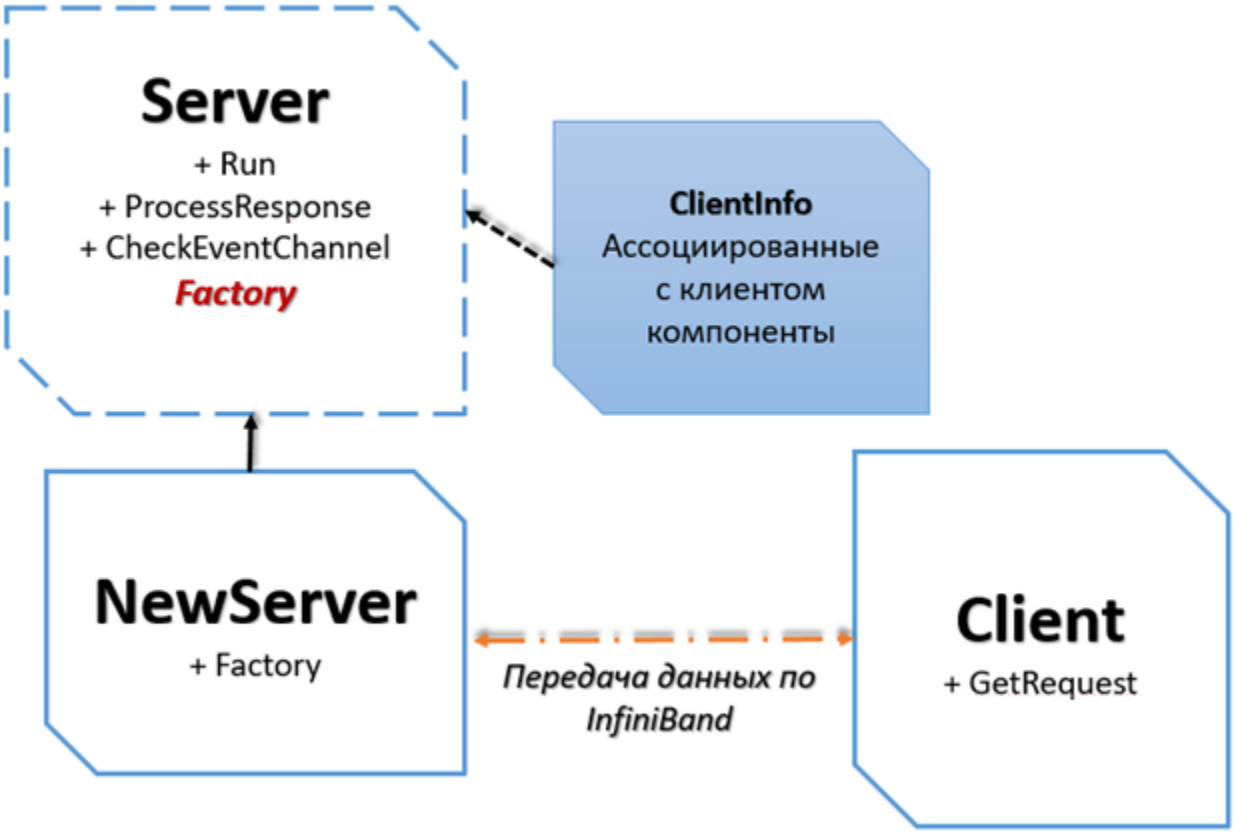

Рисунок 6. *Диаграмма отношений между классами, реализованными в разрабатываемом программном интерфейсе.*

Решение же второй подзадачи становится обязанностью пользователя программного интерфейса. С помощью механизма перегрузки языка C++ появляется возможность менять функциональность сервера в зависимости от задач, которые он должен выполнять – обрабатывать запросы, быть посредником в соединении, отправлять

запросы к базе данных и другие. Для этого необходимо реализовать, например, класс NewServer (Рисунок 5), который будет наследником класса Server с перегрузкой функции Factory:

```
class NewServer : Server
{
  NewServer(char* port) : Server(port) {}
  protected: void Factory (char* result, uint32_t length) { Функциональность сервера }
};
```

В итоге с помощью низкоуровневых интерфейсов программирования ibverbs и rdmacm был реализован более высокоуровневый интерфейс “ibv_communication.h”, построенный на следующей идее.

Экземпляр класса Client выступает в качестве агента для отправки RDMA-запросов. С помощью реализованной функциональности узел-клиент может отправлять любые данные – все байты будут упакованы в массив типа uint32_t и отправлены на сервер. Создавая экземпляры класса Client, можно устанавливать необходимое количество новых соединений, изолируя свои запросы от запросов, выполняемых другими Client экземплярами на сервере.

Экземпляр Server класса создать нельзя, поскольку он является абстрактным, поэтому нужно реализовать наследника, в котором будет перегружена необходимым пользователю образом всего одна функция – Factory, принимающая полученные от клиента данные и изменяющая их согласно предполагаемой функциональности. Также пользователь сам определяет необходимую функциональность сервера в виде отправления ответов клиенту. Для включения возможности неотвечающего сервера, необходимо передать значение true как аргумент в функцию ProcessResponce.

Разработанные классы инкапсулируют трудности низкоуровневого программирования с помощью IB verbs и RDMA CM. Они скрывают интерфейсом все тонкости и особенности от пользователя, разрабатывающего свое приложение, которое использует передачу данных по сети InfiniBand. Функциональность реализованного программного интерфейса аналогична организации запросов к http-серверу, использующейся в сети Интернет.

7. Тестирование и анализ исследования программного интерфейса передачи данных между узлами суперкомпьютера

Одной из поставленных в работе задач является запуск и тестирование пропускной способности канала сообщений разработанного интерфейса с технологией InfiniBand на узлах суперкомпьютера. Для этого был создан класс NewServer, который в качестве функции Factory не реализует никакую функциональность, чтобы снизить влияние накладных расходов на обработку данных, т.е. чтобы измерить пропускную скорость канала. В программе, реализующей сторону клиента, производилась передача данных с помощью функций GetRequest в цикле. За все время тестирования для различных размеров сообщений производился запуск 500 миллионов итераций, чтобы сгладить все случайные выбросы. Размер сообщений варьировался от 64 байт до 4096 байт.

Процесс тестирования заключался в фиксировании начальной отметки времени в программе клиентской стороны, после чего происходил цикл итераций отправления данных, а затем производилась повторная фиксация конечной отметки времени. Разница между начальной и конечной отметкой показывает количество времени, затраченное на передачу. Для измерения времени использовалась функция clock_gettime(CLOCK_REALTIME) из библиотеки time.h.

Серверная сторона в свою очередь ведет подсчет полученных пакетов. Поскольку для передачи используется транспортный режим UD, некоторая часть пакетов теряется – примерно 2% от 500 миллионов пакетов. Это объясняется тем, что серверная сторона не успевает поставить запрос на получение данных в канал сообщений, когда клиентская сторона уже отправляет данные. Пропускная способность канала разработанного интерфейса измерялся как количество байт, переданное за весь цикл, деленное на время, которое этот цикл длился.

На рисунке 7 показана зависимость от размера передаваемого пакета пропускной способности канала сообщений InfiniBand в разработанном интерфейсе. Горизонтальная ось показывает размер пакета в байтах, а вертикальная – вычисленную пропускную способность в Гбит/секунду.

Рисунок 7. *Зависимость пропускной способности канала передачи данных по сети InfiniBand от размера пакета при запуске разработанного программного интерфейса на узлах суперкомпьютера «Ломоносов-2».*

Тестирование показывает, что полученная пропускная способность канала сообщений приближается к 12,5 Гбит/секунду. При изменении размера передаваемого пакета до 2048 байт потери пакетов практически не наблюдаются – около 2% от всех переданных пакетов за весь цикл. Однако при увеличении размера после 2048 процент потерь резко возрастает до 50%, что говорит о достижении максимальной пропускной способности канала. Исследование показывает, что выгоднее передавать данные размером 2048 байт при высокой нагрузке на сеть, и это стоит учитывать при использовании интерфейса для построения системы мониторинга суперкомпьютера.

8. Интеграция разработанного программного интерфейса в систему мониторинга суперкомпьютера DiMMon

В системе мониторинга используются модули с различной функциональностью, например, модуль, который получает данные о процентах загрузки производительности ядра вычислительного узла, или модуль, который выводит переданные ему данные на экран. Взаимодействие таких модулей между разными агентами мониторинга невозможно без еще одного важного модуля – модуля передачи данных. Как уже говорилось, этот

модуль состоит из двух типов узлов – принимающего данные с входа и передающего полученные после передачи данные на выход. Таковыми являются узлы Send и Recv соответственно. В системе DiMMon такая передача данных между агентами реализована с помощью сокетов и протокола UDP. Однако в рамках данной научной работы произведена реализация транспортировки данных мониторинга по сети InfiniBand с использованием разработанного программного интерфейса на основе UD.

Поскольку Recv имеет только выходные соединения с другими модулями, система посылает только управляющие сообщения. Однако Send получает как управляющие сообщения, так и данные мониторинга, которые затем отправляются указанному узлу. Управляющие сообщения, которые приходят на узел Send, передают указания о создании соединения с узлом Recv, адрес которого содержится в конфигурационном файле или может быть задан статически.

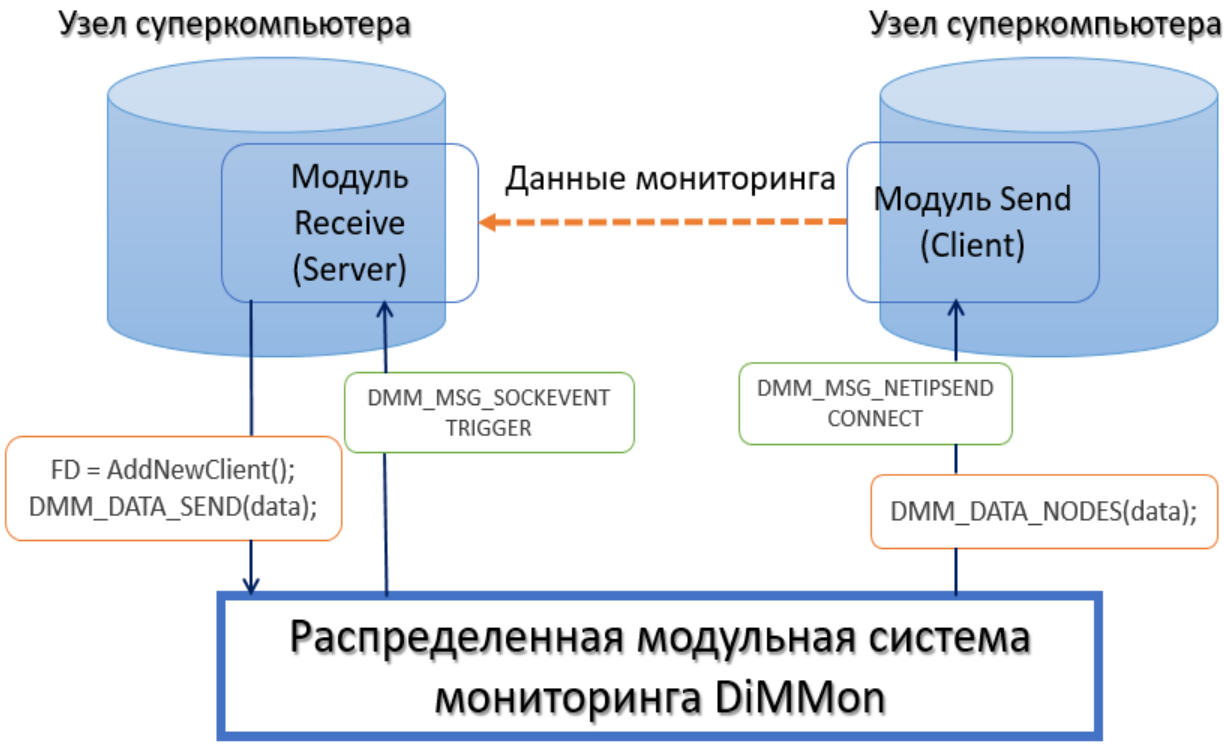

Рисунок 8. *Реализация взаимодействия системы мониторинга DiMMon и разработанных программных модулей для организации передачи данных по сети InfiniBand.*

Для реализации такого взаимодействия с использованием разработанного интерфейса узел Send создает экземпляр класса Client при получении управляющей информации о создании соединения. Благодаря удобной функциональности интерфейса передача данных мониторинга происходит с помощью вызова одной функции – GetRequest, которая принимает указатель на массив байт и его длину. С помощью

управляющих сообщений система DiMMon создает экземпляр клиента – при получении события DMM_MSG_NETIPSEND_CONNECT, который устанавливает соединение с сервером, адрес которого принимается в виде строки из конфигурационного файла, после чего преобразуется в необходимый для программы вид. При возникновении новых данных на модулях, которые имеют соединения с узлом Send, система вызывает функцию send_rcvdata (рис.8), затем происходит проверка подключения к серверу или наличия сбоя, который мог возникнуть, например, при парсинге адреса из конфигурационного файла из-за некорректного ввода. При запуске функции send_rcvdata системой узел получает данные, которые лежат во входящем соединении (hook), с помощью макроса системы DMM_DATA_NODES(data). Следующим шагом вычисляется длина сообщения, которое будет передано в функцию GetRequest вместе с указателем на эти данные, как разница между первым и последним байтом сообщения. Упрощенная часть кода выглядит следующим образом:

```
static int send_rcvdata(dmm_hook_p hook, dmm_data_p data)
{
    pvt = (struct pvt_data *)DMM_NODE_PRIVATE(DMM_HOOK_NODE(hook));
    if (!(pvt->flags & DMM_NETIPSEND_CONNECTED)) {
        err = ENOTCONN;
        goto error;
    }
    dn = DMM_DATA_NODES(data);
    for ( ; !DMM_DN_ISEND(dn); dn = DMM_DN_NEXT(dn))
        len = (char *)dn - (char *)DMM_DATA_NODES(data) + sizeof(struct dmm_datanode);
    pvt->client->GetRequest(data->da_nodes, NULL, len);
}
```

В указанной части кода используется вспомогательная структура pvt, которая хранит указатели на входящие соединения, созданный экземпляр клиента, флаги и другую техническую информацию.

Узел Recv реализуется немного сложнее. Этот тип узла не имеет возможности получать данные от системы, поэтому DiMMon управляет серверной частью только с помощью управляющих сообщений в отличие от клиентской. При получении управляющей информации в виде события DMM_MSG_NETIPRECV_CREATESOCK о переходе в состояние ожидания передачи данных создается экземпляр класса NewServer, который пользователь должен реализовать с помощью наследования от основного класса

Server. В конструктор экземпляра класса передается номер порта, на котором сервер будет ожидать подключения клиентов. По аналогии с узлом Send происходит парсинг строки адреса из конфигурационного файла. При обработке управляющей информации происходит еще одна важная часть взаимодействия узла и системы – происходит вызов макроса DMM_MSG_DATA, с помощью которого происходит передача агенту файловый дескриптора EventChannel для отслеживания его на наличие подключения новых клиентов по принципу epoll. Макрос DMM_MSG_CREATE позволяет произвести подписку на такие события от системы, а макрос DMM_MSG_SEND_ID оповещает систему о произведенной подписке. В общем виде это выглядит следующим образом:

```
static int process_bind_msg(dmm_node_p node, dmm_msg_p msg)
{
pvt->server = new DmmServer(port, node);
    pvt->fd = pvt->server->GetFd();
    ses = DMM_MSG_CREATE(DMM_NODE_ID(node),
                         DMM_MSG_SOCKEVENTSUBSCRIBE,
                         DMM_MSGTYPE_GENERIC,
                         GET_TOKEN(),
                         sizeof(struct dmm_msg_sockeventsubscribe)
                        );
    DMM_MSG_DATA(ses, struct dmm_msg_sockeventsubscribe)->fd = pvt->fd;
    DMM_MSG_DATA(ses, struct dmm_msg_sockeventsubscribe)->events =
    DMM_SOCKEVENT_IN;
    DMM_MSG_SEND_ID(DMM_NODE_ID(node), ses);
}
```

Управляющее сообщение типа DMM_MSG_SOCKEVENTTRIGGER отправляется системой узлу Recv в двух случаях. В первом случае новый клиент отправил запрос на подключение, поэтому необходимо обработать канал событий подключения вызовом функции CheckEventChannel, которая возвращает файловый дескриптор из канала обмена сообщений CompletionQueue непосредственно с указанным клиентом. Этот случай требует произвести подписку на отслеживание системой этого дескриптора аналогично подписке в канале подключений. Оба вида дескрипторов должны быть неблокирующими, чтобы система могла просматривать их по принципу epoll, и это гарантируется разработанным интерфейсом. Как только на один из дескрипторов придет информация – в случае добавления нового клиента или в случае получения новых данных - система вновь отправит узлу управляющее сообщение, которое будет обработано соответствующим

образом. Во втором случае управляющее сообщение будет отправлено при возникновении данных на файловом дескрипторе канала обмена сообщений с каким-то из уже подключенных клиентов. Это потребует вызова функции *pvt->server->ProcessResponse* с указанием флага неотвечающего сервера. В обоих этих случаях вызывается функция static int process_socket_event(dmm_node_p node, uint32_t events). При получении данных мониторинга в функции ProcessResponse необходимо организовать их корректную передачу на выходящие соединения (out hook) или высвободить данные в случае отсутствия соединений. Это обеспечивается с помощью перегруженной в классе NewServer функции Factory, которая передает полученные данные системе DiMMon в случае ненулевого указателя на исходящее соединение с помощью макроса DMM_DATA_SEND:

```
protected void Factory (char* result, uint32_t byte_len)
{
        dmm_data_p      data;
        dmm_datanode_p  dn;
        dmm_size_t      bytes_recvd = byte_len;
        if (outhook != NULL)
        {
                dn = DMM_DATA_NODES(data);
                memcpy(dn, result, bytes_recvd);
                DMM_DATA_SEND(data, outhook);
                DMM_DATA_UNREF(data);
        }
}
```

В итоге передача данных между модулями превращается в обычное взаимодействие классов Client и Server под управлением агентов системы. Благодаря удобной функциональности разработанного программного интерфейса для передачи данных по InfiniBand он легко интегрируется в систему DiMMon в виде вызовов высокоуровневых функций, работающих с компонентами InfiniBand. Разработанный интерфейс позволяет полностью сохранить изначальную логику работы системы DiMMon, изменив только обращения к сокетам на схожие по выполняемым обязанностям функции созданных экземпляров классов.

9. Тестирование и анализ работы системы мониторинга суперкомпьютера DiMMon с внедренными модулями передачи данных по сети InfiniBand

Для достижения поставленной цели необходимо провести тестирование системы мониторинга DiMMon с внедренными модулями передачи данных по сети InfiniBand. Тестирование работы системы мониторинга проводилось на узлах суперкомпьютера «Ломоносов-2».

При анализе влияния внедренных модулей на работу системы мониторинга использовалась UNIX-команда top, которая выводит список работающих в системе процессов, а также информацию о них. Аналогично предыдущему тестированию программы запускались на выделенных узлах, поэтому на загрузку процессора влиял только один процесс – модуль, передающий данные. Исследование проводилось в зависимости от периода передачи данных системы мониторинга, который регулировался срабатыванием таймера в конфигурационном файле. Таймеру передавалось два числа – количество секунд и количество наносекунд, через которые данные будут переданы. На рисунке 9 представлено сравнительное влияние модулей, передающих данные по сети Ethernet с помощью UDP и по сети InfiniBand с помощью UD. На горизонтальной оси отображена частота передачи данных, измеряющаяся в количестве пакетов, отправленных за секунду. Вертикальная ось показывает загрузку центрального процессора передающего узла суперкомпьютера в процентах, которое выдает команда top.

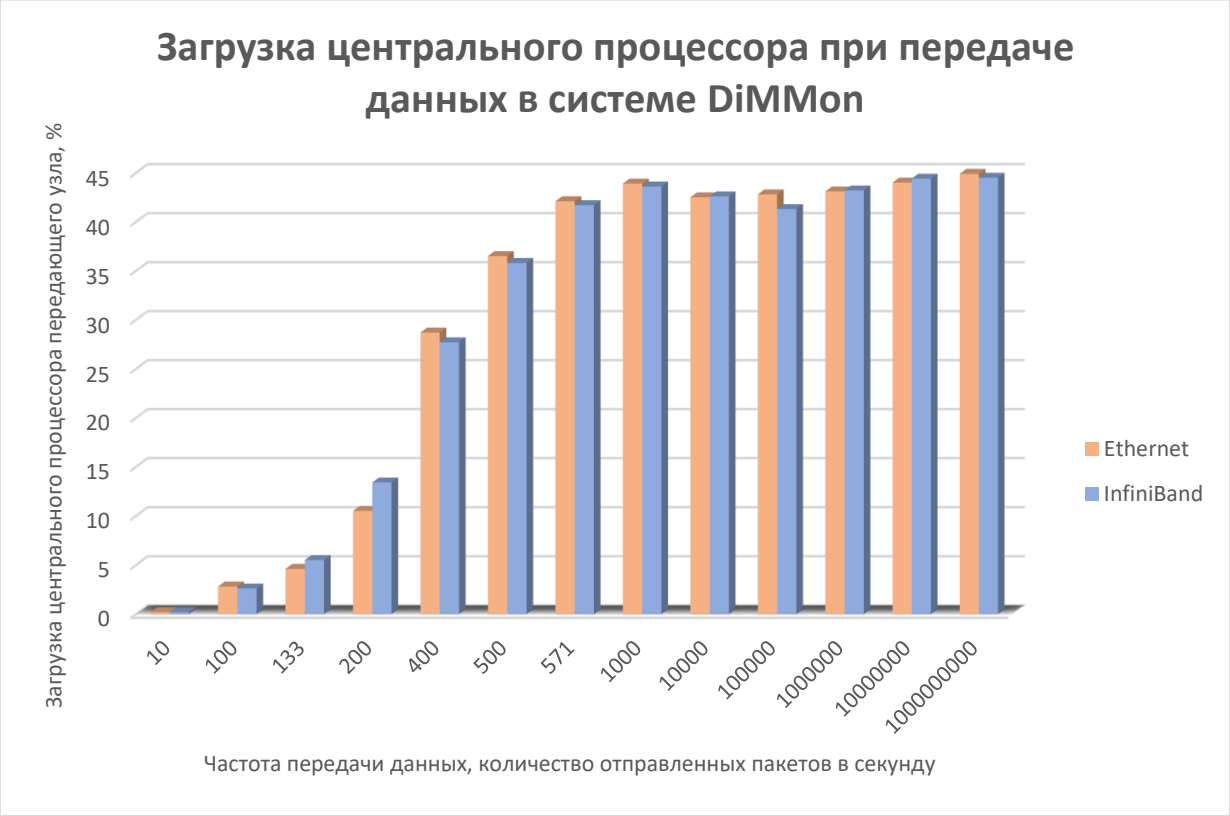

Рисунок 9. *Исследование влияния запущенной на узлах суперкомпьютера «Ломоносов-2» системы мониторинга DiMMon с передачей данных по Ethernet (UDP) и InfiniBand (UD) на загрузку процессора в процентах.*

Тестирование показало, что система DiMMon и с передачей данных по сети Ethernet, и с передачей данных по сети InfiniBand практически одинаково загружают центральный процессор, то есть передача данных не оказывает существенную нагрузку. Это объясняется тем, что команда top показывает загрузку процессора всем процессом, порожденным запуском агента системы DiMMon, а не отдельно модуля передачи данных, то есть процессор нагружает функционирование других модулей запущенного агента. Тем не менее, это показывает, что даже при внедрении сложного интерфейса, инкапсулирующего низкоуровневое программирование на IB verbs и RDMA CM, система мониторинга показывает аналогично хорошие результаты, что и при реализации транспорта с относительно низкоуровневым программированием для UDP.

10. Заключение

В ходе научной работы поставленные задачи были выполнены. В итоге были достигнуты следующие результаты:

- Разработана библиотека, реализующая высокоуровневый интерфейс для передачи данных с использованием технологии InfiniBand с помощью Unreliable Datagram;

- На основе полученной библиотеки разработан модуль передачи данных при помощи InfiniBand для системы мониторинга производительности суперкомпьютера DiMMon;
- Проведено исследование производительности разработанных библиотеки и модулей системы мониторинга на суперкомпьютере «Ломоносов-2». Результаты показывают эффективность использования InfiniBand в качестве транспорта для передачи данных системы мониторинга суперкомпьютера.

11. Дальнейшая работа

Разработанный программный интерфейс может быть использован для передачи данных с помощью технологии InfiniBand не только в системе мониторинга суперкомпьютера, но и в других областях, в которых требуется транспорт данных между приложениями. Дальнейший интерес для исследования представляет внедрение интерфейса в практическое применение, а также изучение масштабируемости программного модуля.

Список литературы

[1] Mellanox Technologies. Информационный документ InfiniBand Solutions 200G HDR, 2019. URL: https://www.mellanox.com/related-docs/whitepapers/WP_Introducing_200G_HDR_InfiniBand_Solutions.pdf.

[2] Список самых производительных систем мира, ноябрь 2020. URL: https://www.top500.org/lists/top500/2020/11/.

[3] Stefanov K. и др. Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon) // 4th International Young Scientist Conference on Computational Science / под ред. Sloot P. и др. Elsevier B.V., 2015. Т. 66. С. 625–634.

[4] Agelastos A. и др. The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications // SC14: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2014. С. 154–165.

[5] InfiniBand Trade Association. InfiniBand Architecture Specification Volume 2, Release 1.3.1, November 2016. URL: http://www.infinibandta.org.

[6] Paul Grun, InfiniBand Trade Association. Introduction to InfiniBand for End Users, 2010. //Industry-Standard Value and Performance for High Performance Computing and the Enterprise, 3855 SW 153rd Drive Beaverton, OR 97006. URL: https://www.mellanox.com/pdf/whitepapers /Intro_to_IB_for_End_Users.pdf. С. 5-11.

[7] Mellanox Technologies. Introduction to InfiniBand, 2003 // Inc. 2900 Stender Way, Santa Clara, CA 95054. URL: https://www.mellanox.com/pdf/whitepapers/IB_Intro_WP_190.pdf. С. 18-19.

[8] Mellanox Technologies. RDMA Aware Networks Programming User Manual Rev 1.7, 2015. //Mellanox Technologies 350 Oakmead Parkway Suite 100 Sunnyvale, CA 94085 U.S.A. URL: https://www.mellanox.com/related-docs/prod_software/RDMA_Aware_Programming_user_ manual.pdf. С.20-21

[9] Gregory Kerr, College of Computer and Information ScienceNortheastern UniversityBoston, MA. Dissecting a Small InfiniBand Application Using theVerbs API, 2011.

[10] GREGORY F. PFISTER, Chapter 42: An Introduction to the InfiniBand Architecture // High Performance Mass Storage and Parallel I/O: Technologies and Applications — Wiley, 2002, ISBN 978-0-471-20809-9. С. 625—627.

[11] ProgrammerSought [сайт]. URL: https://www.programmersought.com/article/5730580112/

[12] RDMAmojo, March 2013 [сайт]. URL: https://www.rdmamojo.com/2013/03/09/ibv_get_cq_event/