Программная платформа передачи интенсивных потоков данных на удаленные суперкомпьютеры*

В.А. Щапов[1,2], Г.Ф. Масич[1,2]

Институт механики сплошных сред УрО РАН[1], Пермский национальный исследовательский политехнический университет[2]

Многие современные экспериментальные установки порождают интенсивные потоки данных, которые необходимо обрабатывать и сохранять. Наличие доступных высокоскоростных сетей передачи данных позволяет использовать для этих целей ресурсы удаленных суперкомпьютерных центров и датацентров, что требует создания программной инфраструктуры для эффективной передачи и распределения данных по вычислителям или узлам хранения данных. В статье приводятся результаты исследования по созданию программной платформы передачи интенсивных потоков данных на удаленные суперкомпьютеры для параллельной обработки.

1. Введение

В последнее время известные проекты в области e-Science затрагивают обработку все больших наборов данных, получаемых от удаленных экспериментальных установок (например, CERN LHC в физике высоких энергий и голландский проект LOFAR в астрономии). Изначально в грид-вычислениях использовались разделяемые между всеми пользователями Интернет TCP/IP сети. Текущий этап развития технологий распределенных вычислений ориентирован на использование национальных и региональных научно-образовательных оптических сетей (например, Geant2 в Европе, Internet2 в США, Initiative GIGA UrB RAS в России). Предмет исследований и разработок текущего времени сфокусирован на решении двух связанных задач: (1) эффективном использовании скоростных (10-100 Гбит/с) протяженных (сотни километров) линий связи и (2) способах организации скоростного ввода/вывода в суперкомпьютер. Так, например, известные проекты Суперкомпьютерного центра в Питтсбурге (Advanced Networking, Three Rivers Optical Exchange, Web10G) направлены на повышение скорости доступа к хранилищам данных и тюнинг TCP протокола [1].

В работе развита принципиально новая идея ввода измеряемых величин в вычислительные узлы удаленного суперкомпьютера по скоростной оптической сети [2]. Эта идея основана на возможности независимой обработки измерений, генерируемых в установках, например, использующих бесконтактные оптические методы измерений (PIV, PTV, PLIF). Апробация разработанных решений выполнялась с использованием научно-образовательной DWDM магистрали (3x10 Гбит/с), которая соединяет установки ИМСС УрО РАН (Пермь) с суперкомпьютерным центром ИММ УрО РАН (Екатеринбург). Перенос вычислений на многопроцессорные системы позволит использовать ресурсоемкие высокоточные алгоритмы, избежать хранения гигантских объемов избыточной информации, обрабатывать измерения в темпе их генерации и проводить эксперименты с обратной связью.

2. Существующее и предлагаемое решения

В задачах обработки больших объемов данных на суперкомпьютерах источником данных обычно является система хранения данных суперкомпьютера. В этом, «классическом режиме» на рис. 1, обработка происходит в три этапа:

1. Загрузка данных в хранилище суперкомпьютера.
2. Обработка данных на суперкомпьютере.
3. Выгрузка результатов обработки с хранилища суперкомпьютера.

* Работа выполнена при поддержке гранта РФФИ №14-07-96001 «Разработка архитектуры распределенного интерконнекта».

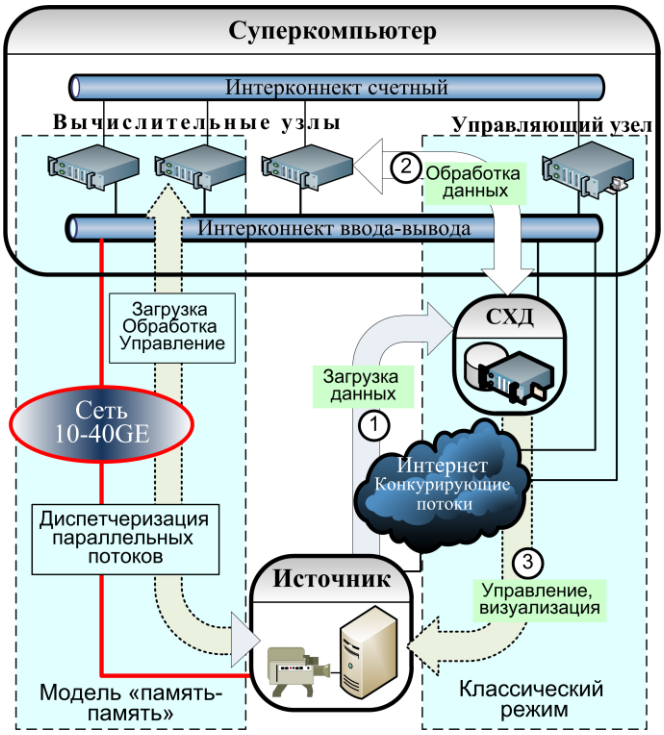

Рис. 1. Существующее (классический режим) и разрабатываемое (память-память) архитектурное решение обработки данных на суперкомпьютере

Загрузка/выгрузка данных в/из хранилища (этапы 1 и 3) и последующая обработка (этап 2) связаны с промежуточными операциями записи/чтения данных в хранилище суперкомпьютера.

Наиболее частым способом обмена данными с хранилищами является использование протоколов передачи файлов, таких как FTP/GridFTP и SCP. Другим направлением данного подхода является прямой доступ к хранилищу данных при помощи протоколов работы с файловой системой, таких как CIFS и NFS/pNFS. Этот способ позволяет подключить хранилище суперкомпьютера к источнику данных как удаленную файловую систему и производить запись данных без использования специализированного программного обеспечения.

Однако на протяженных высокоскоростных линиях связи эти решения работают недостаточно хорошо [3,4]. Общепризнано, что одним из основных источников ухудшения совокупной производительности территориально распределенных высокоскоростных приложений является плохая end-to-end производительность протокола TCP, который по умолчанию присутствует во всех системах. Например, в распределенной системе хранения данных, построенной из 4-х серверов dCache в Перми и Екатеринбурге, при работе по выделенному каналу связи Пермь-Екатеринбург пропускной способностью 10 Гбит/с, скорость записи данных изменялась от 160 Мбит/с при работе в один поток до 3200 Мбит/с при работе в 32 потока [5].

В настоящей работе рассматривается подход, при котором элементы потока данных вводятся в вычислительные узлы и обрабатываются параллельно (модель «память-память» на рис. 1). В этом случае на первый план выходит необходимость обеспечения эффективной передачи данных на вычислительные узлы суперкомпьютера и решение задачи распределения элементов потока по вычислительным узлам. Эта задача решается компонентой диспетчеризации параллельных потоков в модели «память - память». Прототипом предлагаемой модели обработки данных является концепция очередей данных. Она позволяет реализовать параллельность на уровне обработки и передачи данных и перенести процесс диспетчеризации данных с множества вычислительных узлов на отдельный компонент системы – менеджер очередей.

Отличительные особенности предлагаемого решения:

- отказ от промежуточного хранения данных в хранилищах;
- прямой ввод данных в вычислительные узлы;
- параллелизм L4-соединений между оконечными системами.

Параллелизм L4-соединений позволяет решить вышеотмеченную проблему эффективного использования надежных транспортных протоколов, работающих по протяженным скоростным линиям связи. Этим обеспечивается возможность использования в качестве транспортных протоколов (L4 OSI RM) повсеместно используемых протоколов TCP и UDP/UDT.

3. Модель обработки потока данных в распределенных системах

Во многих задачах по обработке потоков данных, в том числе в задаче по обработке потока данных в эксперименте PIV, возможно представить исходный поток данных в виде последовательности отдельных сообщений (измерений), которые могут обрабатываться прикладными алгоритмами независимо друг от друга.

Нами предлагается изображенная на рис.2 модель обработки потока данных [6]:

- Представление процесса обработки потока данных в виде трех последовательных стадий (уровней на рис.2):
  - ✓ генерация Источником исходных данных,
  - ✓ распределение Менеджером исходных данных по обработчикам,
  - ✓ обработка данных прикладными алгоритмами;
- Представление потока данных от Источника в виде единой очереди сообщений.
- Распределение сообщений Менеджером по вычислительным узлам по запросам от вычислительных узлов в порядке FIFO (First In, First Out).

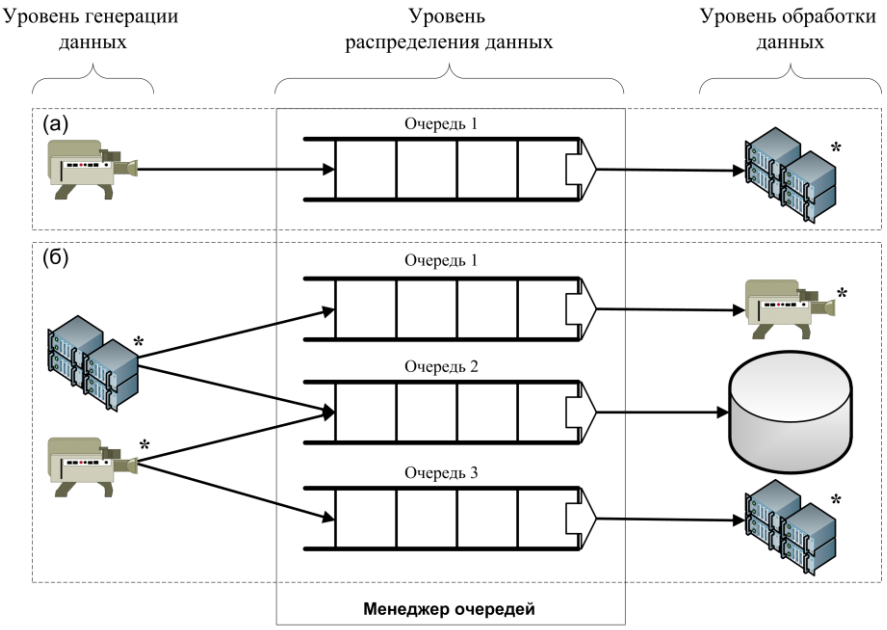

Рис. 2. Модель обработки потока данных

Особенности и гибкость предложенной модели. Между уровнями данные передаются только в одну сторону, от уровня генерации через уровень распределения на уровень обработки данных. В общем случае одно устройство или прикладная программа обработки может находиться одновременно и на уровне генерации данных, и на уровне обработки. Например, расчетное приложение с точки зрения получения исходных данных будет располагаться на уровне обработки данных, а с точки зрения передачи результатов расчета – на уровне генерации.

На уровне генерации данных происходит появление новых данных, требующих передачи на обработку. Это может быть экспериментальная установка, которая генерирует экспериментальные данные или алгоритм расчета, результаты которого должны быть сохранены.

Уровень обработки данных занимается решением задач расчета исходных данных с применением каких-либо прикладных алгоритмов, сохранением данных в высокопроизводительных хранилищах или на большом количестве локальных дисков и т.д.

Уровень распределения данных занимается размещением сообщений, полученных от приложений уровня генерации данных в одной или нескольких очередях менеджера очередей, и передачей данных из очередей приложениям уровня обработки в ответ на их запросы.

Возможность помещать один блок данных сразу в несколько очередей позволяет решать задачу сохранения передаваемых данных, при этом одна очередь используется для передачи данных расчетным приложениям, а вторая – приложениям, отвечающим за сохранение данных (рис. 2, б). Применение описанной модели дает принципиально новые возможности проведения экспериментальных исследований и организации распределенной обработки потоков данных и позволяет:

- Обрабатывать данные от источника и управлять источником данных в реальном времени (рис. 2, б).
- Сохранять (при необходимости) данные от источника и/или результаты счета в системах хранения для последующей обработки или интерпретации результатов счета.
- Избежать необходимости синхронизации между вычислителями на стороне суперкомпьютера.
- Автоматически балансировать нагрузку по вычислительным узлам.
- Изменять число используемых вычислительных узлов во время обработки данных прикладными программами.
- Использовать вычислительную мощность нескольких суперкомпьютеров.
- Избежать потери данных в случае выхода из строя одного или нескольких вычислительных узлов путем сохранения сообщения в буфере менеджера очередей до подтверждения его успешной обработки.

В предлагаемой модели менеджер очередей необходимо располагать как можно ближе к источнику данных (10-1000м) для устранения проблем транспортных протоколов в соединении источник - менеджер. Эффективность передачи данных от менеджера на уровень обработки, который состоит из множества удаленных вычислителей, достигается следующими способами.

1. Параллелизм передачи данных (рис. 3) для устранения проблем транспортных протоколов с обратной связью.
2. Собственные прикладные протоколы взаимодействия оконечных систем и стратегии управления потоком в них.
3. Алгоритм диспетчеризации параллельных потоков, обеспечивающий автоматическое распределение нагрузки по вычислительным узлам.
4. Настройки параметров сетевого стека операционных систем.

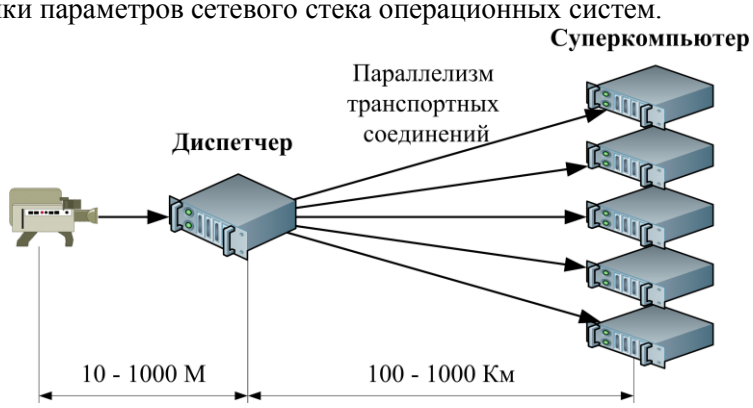

Рис. 3. Параллелизм передачи данных по скоростным протяженным каналам связи

4. Программная инфраструктура распределенной обработки данных

Анализ существующих технологий работы с очередями в распределенных системах (ZeroMQ, протокол AMQP и реализация RabbitMQ) показал, что ни одна из них не удовлетворяет предъявляемым требованиям полностью (скорость работы, простота использования, воз-

можности использовать сторонние транспортные протоколы). В связи с этим было принято решение:

- разработать собственный прикладной протокол передачи данных между уровнями системы SciMP;
- разработать собственную реализацию сервера очередей;
- разработать управляющее программное обеспечение;
- разработать клиентские библиотеки, упрощающие разработку пользовательских приложений – загрузчиков и обработчиков данных.

4.1 Протокол SciMP

Разработанный протокол SciMP является интерактивным протоколом прикладного уровня (рис. 4), работающим по схеме запрос-ответ [7]. Протокол SciMP может использовать в качестве протокола транспортного уровня любой надежный потоковый протокол передачи данных. Текущая реализация поддерживает транспортные протоколы TCP и UDT. Протокол SciMP рассчитан на передачу именованных блоков бинарных данных. В одном пакете может быть передано от 0 до 65535 блоков размером не более 4 Гбайт каждый, при этом сохранение порядка следования блоков не гарантируется. Длина имени блока ограничена 255 байтами. Формат пакета протокола SciMP приведен на рисунке 5.

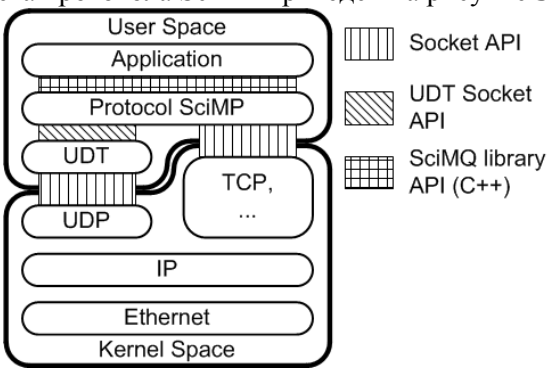

Рис. 4. Положение протокола SciMP в стеке сетевых протоколов

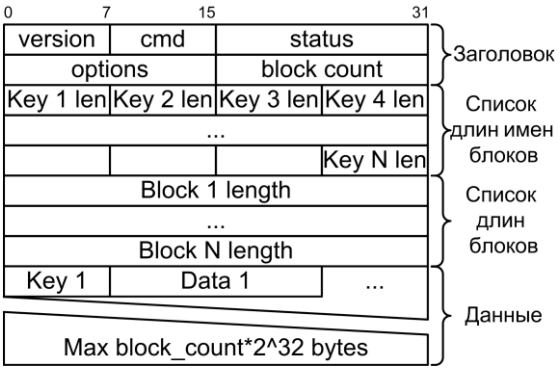

Рис. 5. Формат пакета протокола SciMP

Разработка собственного протокола прикладного уровня обоснована необходимостью работы при скоростях передачи данных более 10 Гбит/с. В этих условиях время, затрачиваемое на разбор форматов сложных протоколов, является существенным и может значительно увеличить системные требования программного комплекса. Протокол SciMP разрабатывался с целью обеспечить простоту разбора формата и минимизировать копирования данных внутри приложения, при этом, не потеряв возможности передавать несколько различных блоков внутри одного пакета. Для достижения этих целей после заголовка в пакете передается список размеров имен и данных для каждого блока. После получения этого списка приложение может выделять память непосредственно под каждый блок данных и проводить дальнейшие чтения из сетевого сокета сразу в целевые области памяти, которые в дальнейшем будут обрабатываться или передаваться в ответ на запросы клиентов.

4.2 Middleware

Предложенная модель обработки потока данных в распределенных системах реализована в виде комплекса программного обеспечения SciMQ [8]. Логическая схема взаимодействия компонент комплекса программного обеспечения показана на рисунке 6.

Основу системы составляет программное обеспечение уровня распределения данных – сервер очередей, в паре с которым работают приложения уровней генерации и обработки данных (оконечных систем). Управление сервером очередей осуществляется при помощи web-интерфейса и интерфейса командной строки.

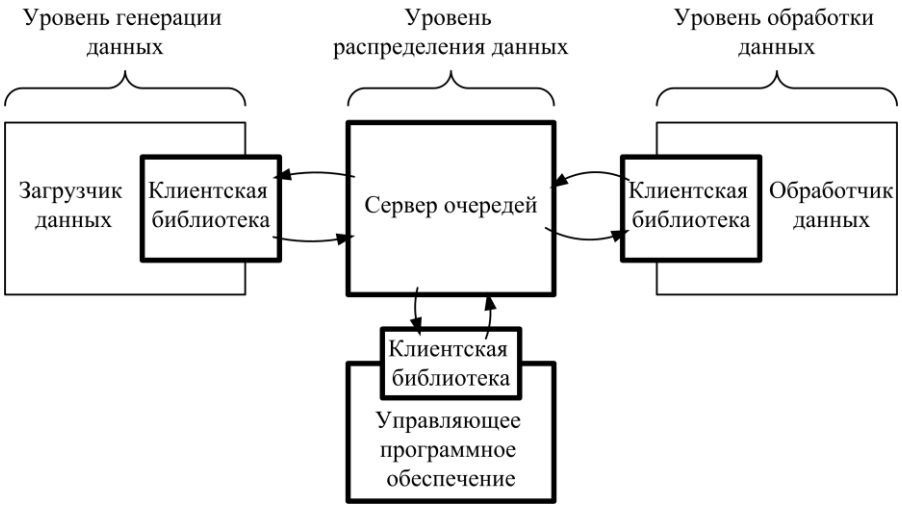

Рис. 6. Логическая схема взаимодействия компонент комплекса программного обеспечения

Приложения оконечных систем взаимодействуют с сервером очередей по сети с использованием протокола SciMP. Для программистов оконечных систем есть два варианта реализации взаимодействия. Первый вариант заключается в самостоятельной реализации протокола SciMP. Этот вариант позволяет добиться максимальной гибкости, в том числе, использовать неподдерживаемые языки программирования.

Второй вариант заключается в использовании C++ API, который предоставляется клиентской библиотекой. Библиотека скрывает от программиста реализацию протокола SciMP и работы с сетью, тем самым позволяя ему сосредоточиться на разработке прикладной задачи, а не реализации протокола взаимодействия. Типовой пример работы с сервером очередей с использованием клиентской библиотеки показан на рисунке 7.

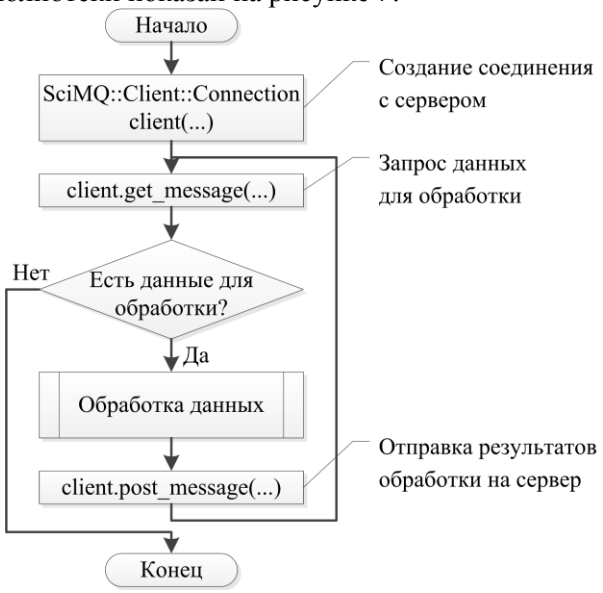

Рис. 7. Типовой пример работы с сервером очередей с использованием клиентской библиотеки

Все компоненты комплекса программного обеспечения разработаны на языке программирования C++. Выбор языка C++ обусловлен как техническими, так и прикладными требованиями. Прикладным требованием является необходимость предоставить конечным пользователям системы API, реализованный на языке C++. С технической точки зрения язык C++ позволяет разрабатывать приложения с использованием высокоуровневого объектно-ориентированного подхода и одновременно обеспечивать низкоуровневый контроль за используемыми ресурсами, в первую очередь ресурсами оперативной памяти, благодаря поддержке ручного управления

памятью. Контроль потребления и освобождения ресурсов особенно важен в случае, когда объемы передаваемых данных существенно больше, чем доступная оперативная память и скорости передачи данных приближаются к пределу возможностей сетевых подсистем, так как позволяют исключить ненужные копирования данных или обнуление блоков памяти перед использованием.

Для уменьшения сложности разработки программного обеспечения требовалось выбрать библиотеки, предоставляющее кроссплатформенный API для работы с сетевыми сокетами и вводом/выводом. В результате анализа были выбраны библиотеки Boost, как обладающие наиболее широкими возможностями, в частности библиотека Boost.Asio, которая предоставляет кроссплатформенный интерфейс для работы с сетевыми сокетами и вводом/выводом с использованием низкоуровневых возможностей, оптимальных для данной платформы. Например, в операционной системе Windows используются механизмы overlapped I/O и I/O completion ports для выполнения асинхронных операций с сокетами, а на платформе Linux механизм epoll.

В современных операционных системах существует два основных вида работы с сетевыми сокетами: это блокирующие (синхронные) и неблокирующие (асинхронные) сокеты. Блокирующие сокеты приостанавливают работу приложения на время выполнения сетевых операций, а неблокирующие сразу возвращают управление приложению и дают механизмы для определения факта завершения операции и ее успешности. Основным способом работы с неблокирующими сокетами является использование механизма событийно-ориентированного ввода/вывода, когда приложение после запуска необходимых неблокирующих операций на множестве активных сокетов вызывает функцию ожидания определенного набора событий. Анализ показывает: событийно-ориентированная модель становится тем эффективнее, чем меньше время обработки данных по сравнению со временем их передачи по сети и, чем больше количество соединений, обрабатываемых приложением. Так как менеджер очередей не выполняет сложной обработки сообщений, а занимается только их диспетчеризацией, то время, затрачиваемое на эти операции, существенно меньше, чем время типичных сетевых операций. Таким образом, для менеджера очередей была выбрана модель неблокирующего, событийно-ориентированного ввода/вывода. В клиентском программном обеспечении было принято решение использовать синхронный механизм ввода/вывода, так как в клиентском программном обеспечении, в общем случае, необходимо работать только с одним соединением с сервером очередей.

Сервер очередей является ядром всего комплекса программного обеспечения и реализует основные функциональные возможности системы. Его задачей является управление сообщениями в очередях, включая их промежуточное хранение, распределение по запросам и прием/передачу от внешних систем. Сервер очередей состоит из четырех модулей: управляющего модуля, модуля очередей, фоновых процессов и сетевого взаимодействия. Все модули, за исключением модуля очередей, работают по событийно-ориентированному принципу с использованием технологий Boost.Asio.

Управляющий модуль отвечает за начальную инициализацию сервера, загрузку параметров конфигурации, запуск и остановку остальных модулей, обработку сигналов операционной системы, таких, как сигналы немедленного и плавного завершения и сигнал для переоткрытия файлов журналов.

Модуль очередей управляет хранением очередей в памяти сервера. Каждая очередь имеет свое уникальное имя и может быть с гарантией обработки или без гарантии обработки. Если очередь без гарантии обработки, то элемент удаляется из очереди сразу же после отправки клиенту. Если очередь с гарантией обработки, то, после передачи клиенту, элемент помещается во временный список, из которого удаляется после получения подтверждения обработки или перемещается в основную очередь, если в течение задаваемого пользователем таймаута подтверждение не было получено. Однако использование таких очередей повышает требования к доступной оперативной памяти, так как данные приходится хранить дольше.

Модуль фоновых процессов предназначен для выполнения низкоприоритетных отложенных задач, таких, как отслеживание таймаутов в очередях с гарантией обработки данных и отслеживание необходимости выгрузки данных из оперативной памяти на диски при достижении лимитов потребления памяти.

Модуль сетевого взаимодействия отвечает за реализацию взаимодействия с клиентами по сети, то есть непосредственно реализует функциональность протокола SciMP. Он представляет собой пул потоков, в каждом из которых работает локальный для потока объект-диспетчер событий boost::asio::io_service. Для возможности обработки нескольких клиентских соединений одним потоком операционной системы применяются неблокирующие сокеты, асинхронные операции и зависимые от операционной системы технологии мультиплексирования, которые реализуются внутри библиотеки Boost.Asio. Каждому соединению с клиентом в момент его создания назначается один из диспетчеров событий io_service, который будет обслуживать все асинхронные операции в этом соединении. Выбор такой архитектуры вызван тем, что операционная система Linux не поддерживает параллельное ожидание событий на одном наборе сокетов из нескольких потоков приложения, поэтому для распараллеливания обработки событий на несколько ядер клиентские сокеты распределяются по разным потокам.

Клиентская библиотека предназначена для упрощения написания прикладных приложений, так как снимает с программиста необходимость написания собственной реализации протокола SciMP. Она предоставляет прикладному программисту синхронный API для взаимодействия с сервером очередей и скрывает все нюансы реализации протокола SciMP и работы с сетью.

Управляющее программное обеспечение предназначено для выполнения задач администрирования и тестирования комплекса программного обеспечения. Оно позволяет выполнять операции по созданию, удалению и модификации очередей и генерировать, получать или перенаправлять потоки данных между очередями, что позволяет проводить нагрузочные тестирования комплекса программного обеспечения для оптимизации настроек сетевых подсистем компьютеров, на которых работают компоненты комплекса программного обеспечения.

5. Измерения

5.1 Инфраструктура

Экспериментальные исследования производились с использованием инфраструктуры проектов «Инициатива GIGA UrB RAS» [9] и «Распределенный PIV» [2]. На рисунке 8 показан фрагмент инфраструктуры, задействованный в исследовании.

Экспериментальная установка PIV (Пермь, ИМСС УрО РАН) соединяется с суперкомпьютером «Уран» и системой хранения данных (СХД) EMC Celerra NS-480 (Екатеринбург, ИММ УрО РАН) по Ethernet технологии на скорости 1-10 Гбит/с. В оконечных системах используется стек протоколов TCP/IP. Каналы связи созданы посредством системы спектрального уплотнения (DWDM) и Ethernet коммутирующего оборудования (рис. 8). DWDM участок Пермь-Екатеринбург реализован в рамках проекта «Инициатива GIGA UrB RAS» с использованием темного волокна магистральных операторов связи (L=456км). Оптические мультиплексоры на 16 лямбда каналов обеспечивают возможность использования транспондеров со скоростью передачи 10-40 Гбит/с в каждом лямбда-канале. Установленные транспондеры используют три лямбда канала со скоростью потоков по 10,7 Гбит/с (ITU-T G.709) и поддерживают механизм упреждающего исправления ошибок. Транспондеры имеют следующие характеристики портов ввода/вывода для подключения оконечного оборудования: 2 Ethernet порта по 10 Гбит/с (2x10GE) и 8 Ethernet портов по 1 Гбит/с (8x1GE). Такая совокупность портов образует слой гарантированных каналов связи по схеме точка-точка. Оставшиеся 13 лямбд — потенциал масштабирования по 10-40 Гбит/с или когерентной DWDM 100 Гбит/с. Для большей гибкости проводимых исследований в Перми и Екатеринбурге установлены L2 коммутаторы, подключённые к 10GE портам транспондеров. Совокупность портов коммутатора образует слой не гарантированных каналов связи при нагрузке больше 10 Гбит/с. Установленное на площадках в Перми и Екатеринбурге серверное оборудование позволяет проводить исследования в области передачи интенсивных потоков данных в диапазоне скоростей 1-10 Гбит/с по линиям связи длиной 0,1-456-912 км. Существует возможность подмешивания в исследуемые потоки интернет трафика, для исследования влияния конкурирующих потоков.

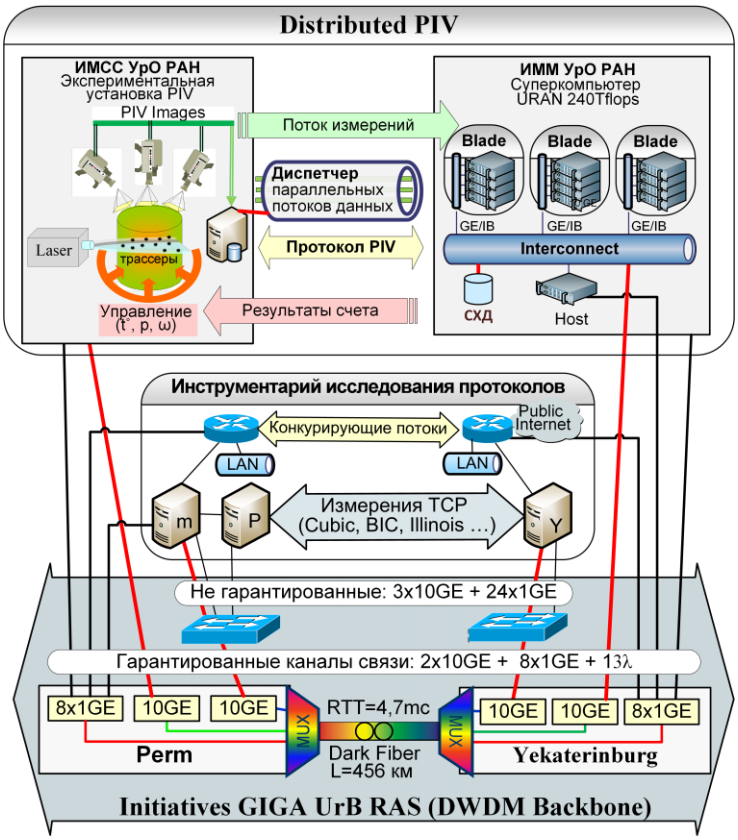

Рис. 8. Инфраструктура

5.2 Тестирование стратегий управления перегрузкой в TCP

Исследование эффективности алгоритмов управления перегрузкой TCP было проведено при помощи специально разработанного ПО, написанного на языке Bash, и приложения Iperf. Тестирование проводилось с использованием выделенного и разделяемого каналов связи на скорости 1 Гбит/с. В разделяемом канале связи помимо исследуемого трафика передавалось до 100 Мбит/с интернет трафика. Результаты тестирования показаны на рисунках 9 и 10.

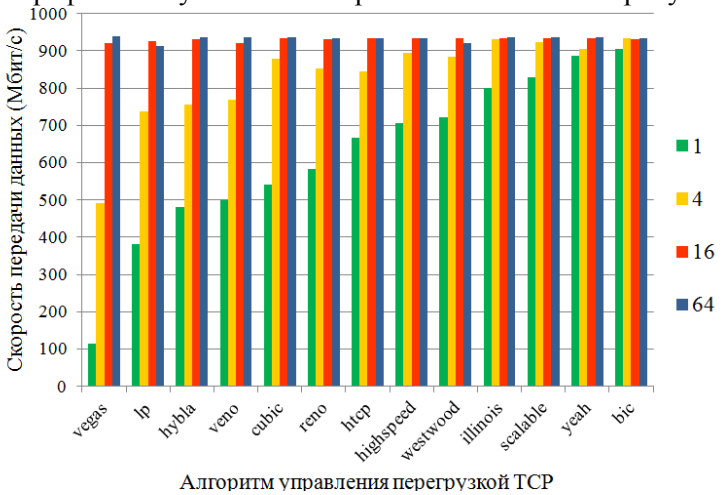

Рис. 9. Сравнение эффективности алгоритмов управления перегрузкой TCP по каналу 1 Гбит/с Пермь-Екатеринбург при наличии фонового трафика (0,1 Гбит/с) и числе параллельных потоков (1, 4, 16, 64)

Из графиков следует, что наличие даже небольшого количества (~ 10%) стороннего трафика существенно снижает эффективность многих алгоритмов управления перегрузкой TCP. Однако большинство алгоритмов достигают совокупной скорости передачи данных близкой

к максимальной пропускной способности канала при числе параллельных потоков в диапазоне от 4 до 16. Поэтому для компонент с большим исходящим трафиком целесообразно повышать эффективность передачи данных для конкретной инфраструктуры следующим образом.

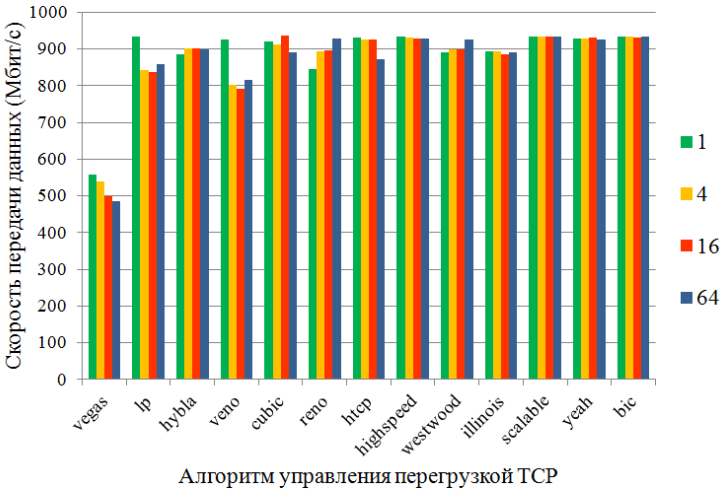

Рис. 10. Сравнение эффективности алгоритмов управления перегрузкой TCP по выделенному каналу 1 Гбит/с Пермь-Екатеринбург при числе параллельных потоков (1, 4, 16, 64)

- Выбрать алгоритм управления перегрузкой. В используемой инфраструктуре проекта «Распределенный PIV» для сервера очередей был выбран алгоритм управления перегрузкой bic, обеспечивающий приемлемую эффективность для выделенного и разделяемого канала передачи данных.
- Настроить размеры буферов, обеспечивающих окно передачи для TCP не меньшее, чем BDP (Bandwidth Delay Product).

Измерения и последующий анализ протокола TCP показали аналогичное влияние существующего во внутренней сети суперкомпьютера трафика на вводимый в вычислительные узлы интенсивный поток данных по каналу с большим BDP. И это несмотря на то, что существующая скорость интерконнекта суперкомпьютера была больше, чем требовалось для экспериментов. Так же подтверждено влияние при большом BDP фазы медленного старта TCP в которой динамика увеличения скорости передачи сдерживает ввод интенсивного потока данных в вычислительные узлы.

5.3 Тестирование производительности разработанного программного обеспечения

Тестирование производительности программного обеспечения проводилось с использованием двух серверов HP ProLiant DL360p Gen8 (2x Intel Xeon CPU E5-2660, 2.20 ГГц, 16 потоков; RAM 128 Гб; операционная система CentOS 6.5), объединенных сетью передачи данных пропускной способностью 10 Гбит/с протяженностью 10м.

На рисунке 11 показаны зависимости пропускной способности разработанного сервера очередей SciMQ и существующего сервера очередей RabbitMQ от размера сообщения и количества параллельных запросов к серверу. Из графиков следует, что при использовании программного обеспечения SciMQ на требуемых на практике размерах сообщений (2-22 Мб), пропускная способность ограничивается в основном доступной скоростью канала связи. Сравнение полученных результатов с результатами тестирования производительности возможных готовых решений на примере RabbitMQ показывает, что разработанное программное обеспечение SciMQ позволяет передавать сообщения требуемых объемов (более 1 Мб) при интенсивности потока данных в 2-4 раза больше, чем RabbitMQ.

5.3 Тестирование стабильности работы системы

Для изучения стабильности работы системы во времени было проведено тестирование, эмулирующее 12-ти часовой эксперимент со скоростью потока исходных данных 8 Гбит/с. Для

этого сервер очередей был подключен на скорости 10 Гбит/с к внутреннему коммутатору интерконнекта суперкомпьютера ИМСС УрО РАН. Поток исходных данных состоял из сообщений размером 16 Мбайт, которые передавались каждые 16 мс. В эксперименте было передано 2700000 сообщений общим объемом 41 Тбайт. За получение данных отвечали 376 процессов, запущенных на суперкомпьютере ИМСС УрО РАН.

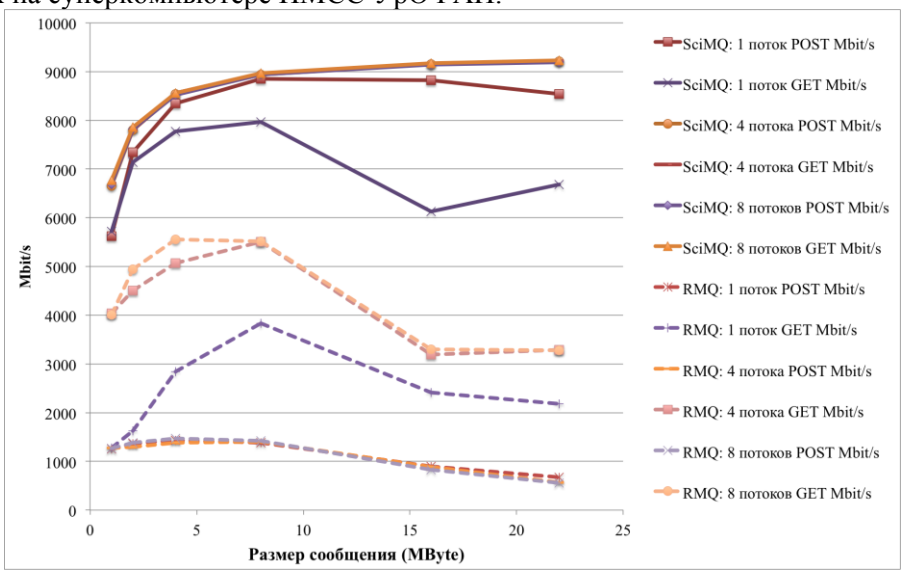

Рис. 11. Графики производительности сервера очередей SciMQ и RabbitMQ

На рисунке 13 показано изменение длины очереди исходных данных (включая число сообщений, находящихся в обработке) и средние интервалы между поступлением исходных сообщений и отправкой сообщений клиентам.

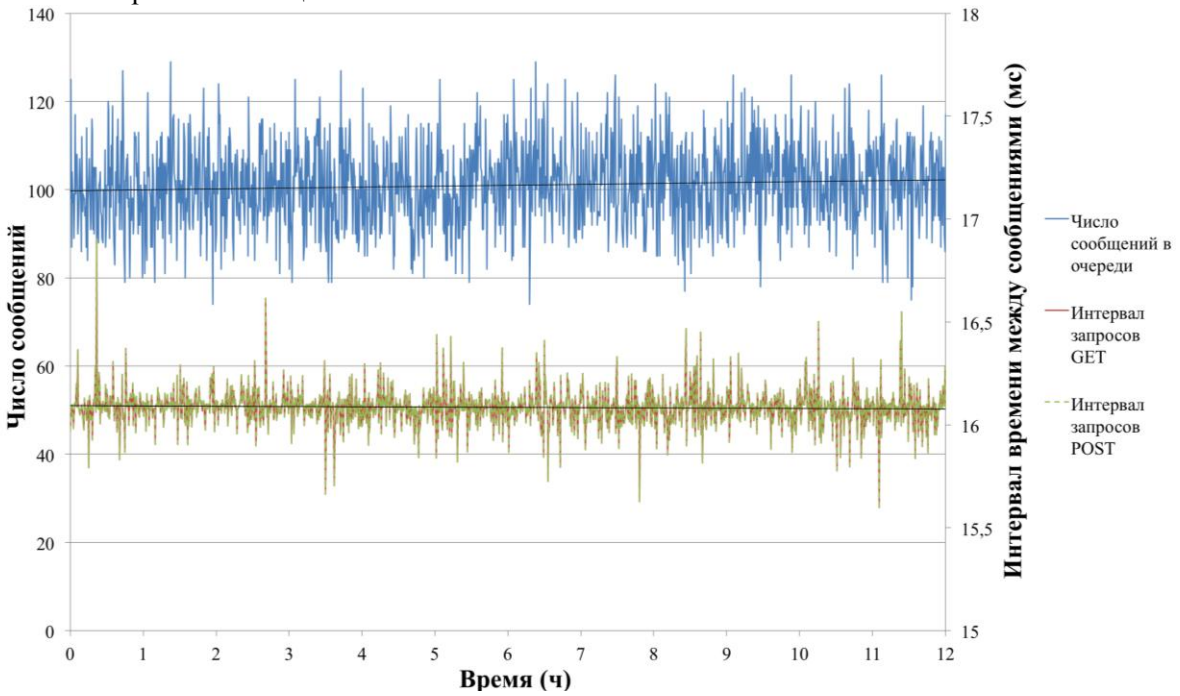

Рис. 12. Графики изменения длины очереди исходных данных (включая число сообщений, находящихся в обработке) и средних интервалов между поступлением и отправкой сообщений клиентам

Из приведенных графиков следует:

- пиковое потребление оперативной памяти сервером очередей было равно 2,03 Гбайт (130 сообщений по 16 Мб в памяти), а среднее – 1,58 Гбайт (101 сообщение по 16 Мб в памяти), что совпадает со статистикой потребления ресурсов, которую ведет программное обеспечение;

- система работает без перегрузок и обеспечивает обработку потока данных без дополнительных задержек и потерь сообщений;
- средние параметры работы системы не меняются во времени;
- программное обеспечение обеспечивает передачу интенсивного потока данных.

6. Заключение

Разработанная технология предоставляет принципиально новый инструмент проведения уникальных физических экспериментов в исследовательских лабораториях и на производстве. Отличительной особенностью созданной технологии является асинхронная модель передачи потока данных от источника в удаленный суперкомпьютер, основанная на идее параллельного прямого ввода структурированного потока данных в вычислительные узлы суперкомпьютера. Скоростные каналы передачи данных позволяют связать место проведения измерений с суперкомпьютерами, устраняя затраты на наращивание вычислительных ресурсов на местах.

Областью применения являются измерительные системы нового поколения, включающие высокотехнологичное измерительное оборудование на местах и высокопроизводительную вычислительную технику в суперкомпьютерных центрах.

Литература

1. Advanced Networking // URL: http://www.psc.edu/index.php/research-programs/advanced-networking (дата обращения: 14.05.2015).
2. Степанов Р.А., Масич А.Г., Масич Г.Ф. Инициативный проект «Распределенный PIV» // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: труды Всероссийской суперкомпьютерной конференции / М. Изд-во МГУ, 2009. С. 360-363.
3. D Hildebrand, M Eshel, R Haskin, P Kovatch, P Andrews, J White Deploying pNFS across the WAN: First Steps in HPC Grid Computing // in Proceedings of the 9th LCI International Conference on High-Performance Clustered Computing, 2008.
4. Weikuan Yu, Rao N.S.V., Wyckoff P., Vetter J.S. Performance of RDMA-capable storage protocols on wide-area network // Petascale Data Storage Workshop, 2008. PDSW '08. 3rd. 2008. P. 1-5.
5. Г.Ф. Масич, А.Г. Масич, В.А. Щапов, С.Р. Латыпов, А.В. Созыкин, Е.Ю.Куклин Архитектура распределенной вычислительной среды УрО РАН // Материалы XIV Международной конференции «Высокопроизводительные параллельные вычисления на кластерных системах (HPC 2014)». Пермь: Издательство ПНИПУ, 2014. С. 281-287.
6. В.А. Щапов, А.Г. Масич, Г.Ф. Масич. Модель потоковой обработки экспериментальных данных в распределенных системах // Вычислительные методы и программирование. 2012. Раздел 2. С. 139-145 ISSN 0507-5386.
7. Vladislav Shchapov, Alexey Masich. Protocol of High Speed Data Transfer from Particle Image Velocimetry System to Supercomputer // Proc. of The 7th International Forum on Strategic Technology (IFOST 2012) September 18-21, Tomsk Polytechnic University, Tomsk, 2012, - Vol.2., P. 653-657 DOI: 10.1109/IFOST.2012.6357642.
8. Щапов В. А. Программная архитектура системы передачи интенсивного потока данных в распределенных системах // Параллельные вычислительные технологии (ПаВТ’2013): труды международной научной конференции (г. Челябинск, 1–5 апреля 2013 г.). Челябинск: Издательский центр ЮУрГУ, 2013. С. 566–576.
9. Масич А.Г., Масич Г.Ф., Матвеенко В.П., Тирон Г.Г. Инициатива GIGA UrB RAS: методология построения и архитектура научно-образовательной оптической магистрали Уральского отделения РАН // Труды Межд. конф. «Математические и информационные технологии» MIT-2011. Белград, 2012. С. 257-265.