УДК 004.271.3

В.А. Щапов[1, 2], Г.Ф. Масич[2]

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

АРХИТЕКТУРА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ОБРАБОТКИ ИНТЕНСИВНОГО ПОТОКА СТРУКТУРИРОВАННЫХ ДАННЫХ НА СУПЕРКОМПЬЮТЕРАХ

Разработана модель и middleware программное обеспечение процесса ввода структурированного потока данных от источника в удаленный суперкомпьютер. Под структурированным потоком понимаются последовательности отдельных сообщений, которые могут параллельно обрабатываться прикладными алгоритмами. Отличительная особенность подхода заключается в прямом вводе сообщений в вычислительные узлы суперкомпьютера в темпе их генерации согласно модели память – память, минуя систему хранения данных. Автоматическая балансировка нагрузки по вычислительным узлам основана на представлении потока данных в виде единой очереди сообщений, которые распределяются менеджером очередей по запросам от вычислительных узлов. Известная проблема эффективной передачи данных по скоростным протяженным оптическим линиям связи решается посредством организации параллельных транспортных соединений между менеджером и вычислительными узлами суперкомпьютера.

Показана тестовая инфраструктура, соединяющая источник потока данных (сервер очередей / менеджер) с суперкомпьютером по оптической линии с варьируемой длиною 0,1–456–900 км на скорости 10 Гбит/с. Приведены результаты тестирования, иллюстрирующие работоспособность и эффективность разработанного middleware SciMQ. Измеренная производительность менеджера очередей SciMQ в 2–4 раза больше измеренной производительности существующей системы управления очередями RabbitMQ. Стабильность работы системы во времени подтверждена путем передачи сообщений размером 16 Мбайт с темпом 20 мс и общим объемом 27,5 Тбайт от менеджера очередей на 376 процессов суперкомпьютера по оптической линии связи длиною 900 км.

Выявлены несколько факторов, отрицательно влияющих на пропускную способность тракта от источника потока данных до удаленного суперкомпьютера (end-to-end скорость). Так, используемая в тестовой инфраструктуре технология агрегации каналов не всегда приводит к пропорциональному числу агрегируемых каналов увеличению скорости. В асимметричной конфигурации сети использование сетевых карт, аппаратно поддерживающих стек протоколов (TCP Offload Engine NIC), приводит к большому числу ошибок и, как следствие, уменьшению скорости передачи.

Разрабатываемые архитектурные решения предоставляют принципиально новый инструмент проведения уникальных физических экспериментов в исследовательских лабораториях и на производстве. Областью применения являются измерительные системы нового поколения, включающие высокотехнологичное измерительное оборудование на местах и высокопроизводительную вычислительную технику в суперкомпьютерных центрах.

Ключевые слова: экспериментальная установка, суперкомпьютер, скоростная оптическая сеть, модель обмена данными, измерение производительности.

V.A. Shchapov [1, 2], G.F. Masich[2]

[1]Perm National Research Polytechnic University, Perm, Russian Federation
[2]Institute of Continuous Media Mechanics, Ural Branch of Russian Academy of Sciences, Perm, Russian Federation

DISTRIBUTED SYSTEM ARCHITECTURE PROCESSING INTENSIVE FLOW OF STRUCTURED DATA ON SUPERCOMPUTERS

The model and middleware software to load structured data stream from a stand to remote supercomputer is developed. Structured data stream is a sequence of messages that can be processed in parallel by user algorithms. A distinctive feature is the direct inputs of messages into compute nodes of the supercomputer at real time, according to the model of "memory-to-memory", bypassing the storage system. Automatic load balancing compute nodes based on the notion of data stream in a single message queue, the queue manager that is distributed at the request of the compute nodes. A well-known problem of efficient data transfer on long high-speed optical links is achieved by the organization of parallel transport links between the data manager and the compute nodes of a supercomputer.

The test infrastructure connecting the source of the data stream (queue server/manager) and supercomputer by the optical line with a variable length 0,1-456-900 km at a speed of 10 Gbit/s is presented. The results of the test, illustrating the efficiency and effectiveness of the developed middleware software SciMQ are presented. The measured performance of the queue manager SciMQ is 2-4 times more than the measured performance of the existing queue manager RabbitMQ. Stability of the system is confirmed by tests: the message transfer size of 16 MB with rate of 20 ms and a total volume of 27.5 terabytes of queue manager 376 processes supercomputer on the optical link length of 900 km.

It is identified several factors that negatively affect performance of the distributed data processing system (end-to-end performance). So used in the testing infrastructure technology aggregation does not always lead to the proportional increases the rate. In the asymmetric network configuration using network cards support hardware protocol stack (TCP Offload Engine NIC), leads to a large number of errors, and the consequent reduction in the transmission rate.

Developed architectural solutions provide a fundamentally new tool of unique physical experiments in research laboratories and industry. Applications include measuring systems of the new generation, including high-tech measuring equipment and high-performance computing to supercomputing centers.

Keywords: experimental stand, supercomputer, high-speed optical network, data communication model, measurement of productivity.

Введение

В предлагаемой работе исследуется процесс ввода интенсивного потока экспериментальных данных в суперкомпьютер и разрабатывается технология параллельной передачи данных. Обычным способом обработки данных является их сохранение на внешних носителях и последующая обработка на компьютере при экспериментальной установке. Однако не все получаемые в экспериментах данные можно обработать на месте, поскольку требуется слишком большая вычислительная мощность и/или система хранения данных. Известный пример распре-

деленной системы – трехуровневая архитектура обработки больших объемов данных, поступающих с LHC (Большой адронный коллайдер).

Классическая схема обработки больших объемов данных на суперкомпьютере имеет три этапа: загрузка данных в хранилище суперкомпьютера, обработка данных на суперкомпьютере, выгрузка результатов обработки из хранилища суперкомпьютера. Нами разрабатывается архитектурное решение, при котором элементы структурированного потока данных вводятся в вычислительные узлы удаленного суперкомпьютера в темпе их генерации и обрабатываются параллельно согласно модели память – память, минуя систему хранения данных. Под структурированным потоком понимаются последовательности отдельных сообщений (измерений), которые могут обрабатываться прикладными алгоритмами независимо друг от друга, и, следовательно, допускается их параллельная обработка.

Одним из примеров такого потока данных являются пары изображений от камер в установках PIV [1], по перемещению трассеров в которых определяется векторное поле.

Другим примером является исследование процессов взаимовлияния в динамической системе «сжимаемая жидкость – деформируемая конструкция», сочетающее численное моделирование с натурным экспериментом (РФФИ № 14-07-96003). Регистрация и предварительный анализ быстропротекающих процессов в рабочей камере выполняются программно-аппаратным комплексом на базе National Instruments NI PXI-1050. Отличительной особенностью этого подхода является соединение математической модели исследуемого процесса на суперкомпьютере с лабораторной установкой, эмулирующей исследуемый процесс.

Однако общепризнанное несоответствие между вычислительной производительностью и компонентами ввода-вывода высокопроизводительных систем текущего поколения сделало ввод-вывод наиболее проблемным местом. При этом одним из основных источников ухудшения совокупной производительности территориально распределенных высокоскоростных приложений является плохая end-to-end производительность повсеместно используемого протокола TCP, а основным механизмом повышения пропускной способности является параллельная передача (GrtidFTP, pNFS).

В этом случае на первый план выходит необходимость обеспечения эффективной передачи данных на вычислительные узлы удаленного суперкомпьютера и решение задачи распределения элементов потока данных по вычислительным узлам.

В этой статье впервые обобщены результаты выполненных исследований по взаимодействию источника интенсивного потока данных с удаленным суперкомпьютером и приведены результаты измерений, иллюстрирующие показатели производительности разработанного программного обеспечения SciMQ.

Модель обработки потока данных

Процесс обработки потока данных можно представить в виде трех стадий:

1) генерация исходных данных;

2) распределение данных по обработчикам;

3) обработка данных прикладными алгоритмами.

Каждой стадии процесса обработки потока данных соответствует одноименный уровень модели: уровень генерации данных, уровень распределения данных, уровень обработки данных (рис. 1).

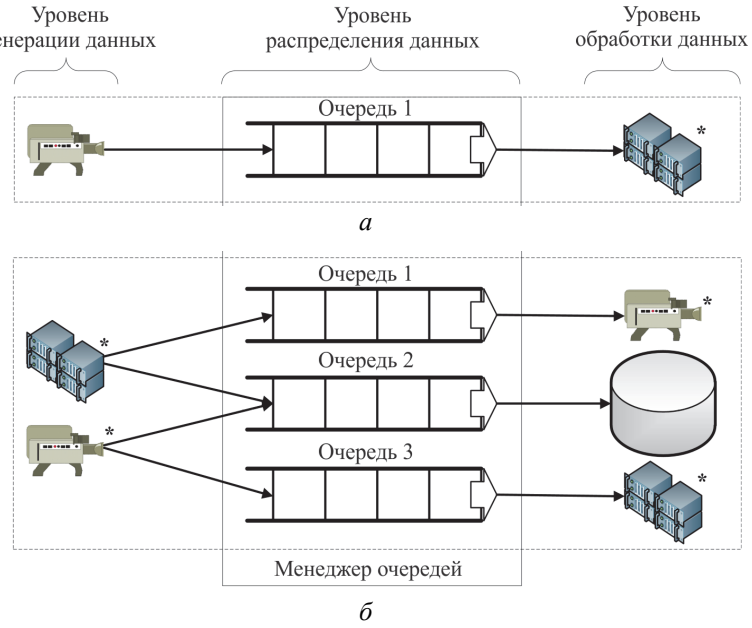

Рис. 1. Процесс передачи потока данных (* – одно устройство может находиться и на уровне генерации данных, и на уровне обработки данных): *а* – размещение в одной очереди; *б* – размещение в нескольких очередях

Между уровнями данные передаются только в одну сторону, от уровня генерации через уровень распределения на уровень обработки данных. В общем случае одно устройство или программа 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 алгоритмов, сохранения данных в высокопроизводительных хранилищах или на большом количестве локальных дисков, передачи данных из очередей в сторонние системы 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. 1, *б*).

Применение описанной модели предоставляет принципиально новые возможности проведения экспериментальных исследований и организации распределенной обработки потоков данных [2]:

– обрабатывать данные от источника и управлять источником данных в реальном времени (онлайн) (см. рис. 1, *б*);

– сохранять (при необходимости) данные от источника и/или результаты счета в системах хранения для последующей обработки или интерпретации результатов;

– 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 в буфере менеджера очередей сообщения до подтверждения успешной его обработки.

В предлагаемой модели менеджер очередей необходимо располагать максимально близко к источнику данных, в этом случае при передаче сообщений от уровня генерации транспортные протоколы будут работать достаточно эффективно, даже без специализированных настроек. В то же время передача данных на уровень обработки, который состоит из множества вычислителей, будет проводиться через большое число параллельных соединений.

Модель распределения сообщений по вычислительным узлам основана на представлении потока данных от источника в виде единой очереди сообщений (рис. 2), которые распределяются менеджером очередей по вычислительным узлам по запросам от вычислительных узлов в порядке FIFO (First In, First Out).

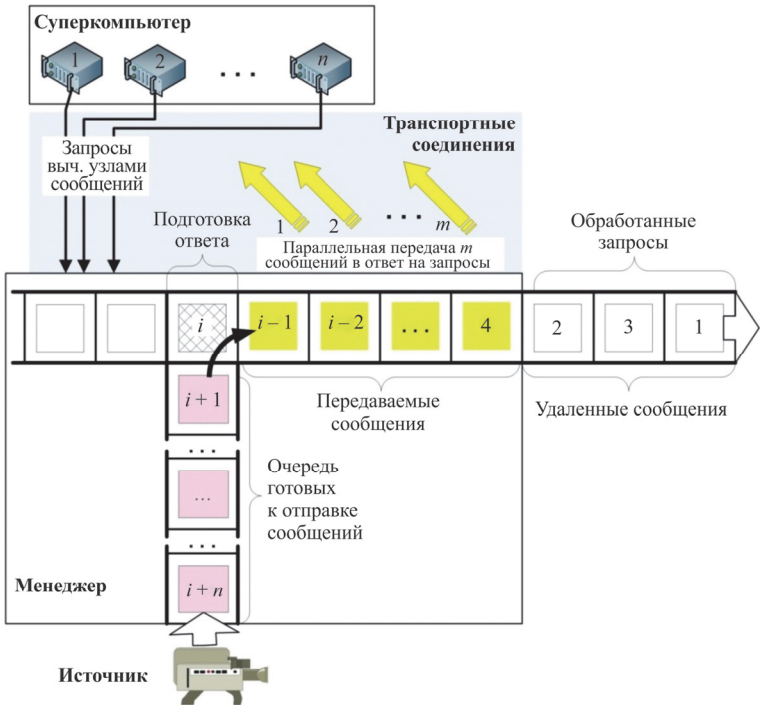

Рис. 2. Модель распределения сообщений по вычислительным узлам

Инициаторами обмена данными являются оконечные системы – источник данных и вычислительные узлы, занимающиеся обработкой данных.

При этом передача данных, в ответ на запросы, может производиться параллельно, что позволяет компенсировать недостатки транспортных протоколов и работать в условиях несимметричных каналов связи, когда скорость потока исходных данных больше, чем скорость подключения к сети передачи данных каждого вычислительного узла.

Выделение задачи диспетчеризации в отдельный слой позволило отказаться от межузлового обмена данными на стороне суперкомпьютера [2].

Существующие решения

Организация систем передачи данных в виде очередей в настоящее время имеет несколько типов практических реализаций:

– системы, предоставляемые в пользование как сервис, доступный через Интернет. Наиболее известным примером данной технологии является сервис Amazon Simple Queue Service [3], входящий в состав облачной платформы Amazon;

– системы управления очередями, например реализующие протокол обмена сообщениями AMQP (Advanced Message Queuing Protocol) [4], такие как RabbitMQ;

– низкоуровневые библиотеки управления очередями и обмена сообщениями, наиболее известной реализацией которых является ZeroMQ [5].

Системы, предоставляемые как сервис, не смогут участвовать в дальнейшем сравнении по причине невозможности их установки в собственную инфраструктуру по требованиям безопасности и производительности.

Системы управления очередями реализуют полный цикл работы с диспетчеризацией и хранением сообщений. Существуют различные реализации протокола, которые используют подобные системы, но наиболее открытым из них является протокол AMQP (Advanced Message Queuing Protocol) – это открытый протокол для передачи сообщений между компонентами распределенной системы. Логика работы протокола декларирует, что обмен данными между компонентами производится через специализированные AMQP-брокеры, которые осуществляют маршрутизацию, возможно, гарантируют доставку, рас-

пределение потоков данных, подписку на нужные типы сообщений и т.д. Наиболее известной реализацией этого стандарта является RabbitMQ – кроссплатформенный менеджер очередей, написанный на языке программирования Erlang.

Среди низкоуровневых библиотек управления очередями и обмена сообщениями наиболее известной является библиотека ZeroMQ. ZeroMQ – это надстройка над интерфейсом сокетов, которая предоставляет возможности по созданию систем, ориентированных на взаимодействие через передачу сообщений.

Оценку производительности разработанного менеджера очередей SciMQ выполним путем ее сравнения с производительностью RabbitMQ.

Программное обеспечение

Предложенная модель обработки потока данных реализована в виде комплекса специализированного программного обеспечения, логическая структура которого приведена на рис. 3.

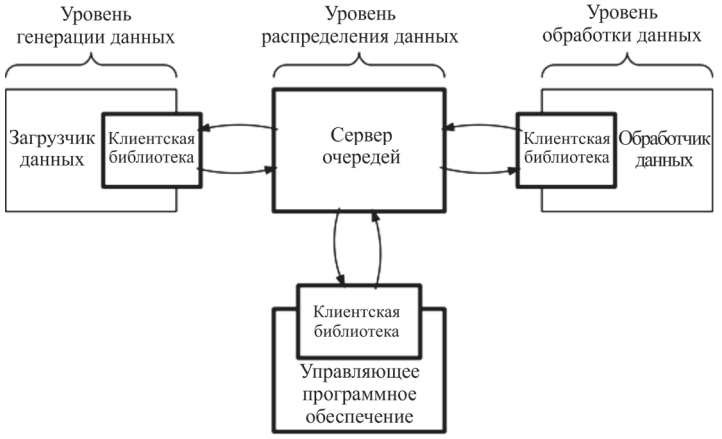

Рис. 3. Логическая структура взаимодействия компонент комплекса программного обеспечения

Менеджер SciMQ, расположенный в сервере очередей, является основной частью реализованного комплекса специализированного программного обеспечения middleware. В паре с менеджером SciMQ работают приложения уровней генерации и обработки данных (оконечных систем) по протоколу SciMP. Приложения оконечных систем взаимодействуют с сервером очередей через API, предоставляемый клиентской библиотекой. Управление сервером очередей осуществляется при помощи веб-интерфейса и интерфейса командной строки.

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 за потреблением и освобождением ресурсов особенно важен в случае, когда объемы передаваемых данных существенно больше, чем доступная оперативная память, и скорости передачи данных приближаются к пределу возможностей процессора, так как позволяют исключить ненужные копирования данных или обнуление блоков памяти перед использованием.

Протокол

Для передачи данных между уровнями системы разработан протокол SciMP, который является интерактивным протоколом прикладного уровня (рис. 4), работающим по схеме запрос-ответ и реализующим идею модели RPC. Разработанный протокол 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 (рис. 5) может быть передано от 0 до 65 535 блоков размером не более 4 Гбайт каждый, при этом сохранение порядка следования блоков не гарантируется. Длина имени блока ограничена 255 байтами.

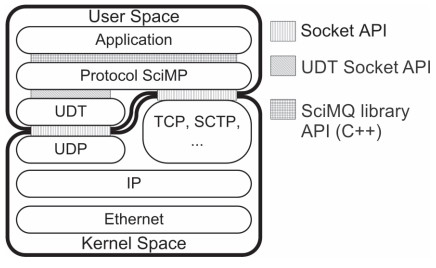

Рис. 4. Положение протокола SciMP в стеке сетевых протоколов

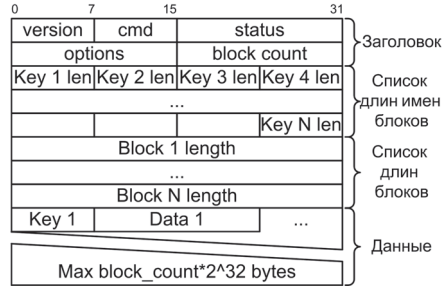

Рис. 5. Формат пакета протокола SciMP

Тестовая инфраструктура

Для апробации разрабатываемых архитектурных решений создана тестовая инфраструктура, соединяющая по оптической среде передачи данных сервер очередей с суперкомпьютером. Использована доступность формирования тестовой петли Пермь – Екатеринбург – Пермь, обеспечивающей канал 10 Гбит/с, протяженностью 900 км между тестовым сервером и кластером ИМСС УрО РАН (рис. 6).

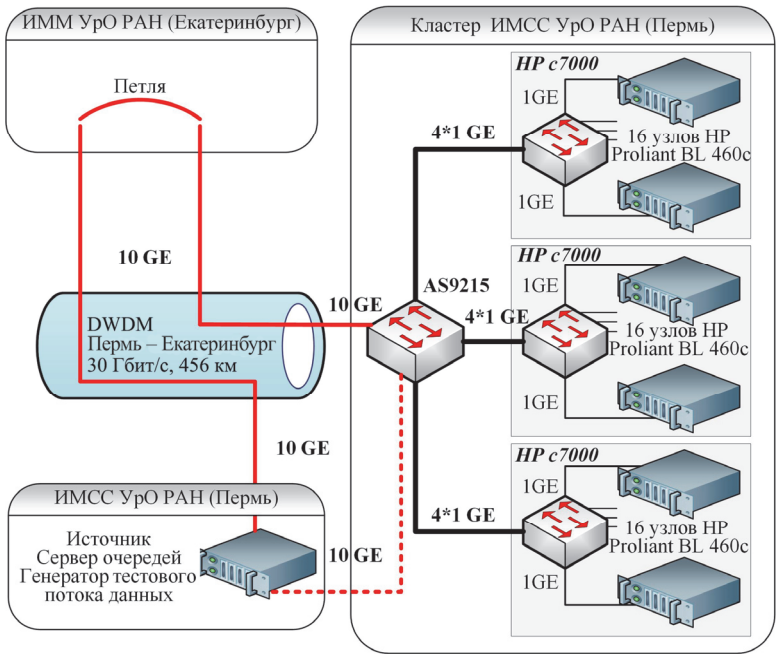

Рис. 6. Схема тестовой инфраструктуры

Сеть. Для соединения источника с кластером используются каналы связи научно-образовательной оптической магистрали со спектральным уплотнением каналов, создаваемой в рамках проекта Initiative GIGA UrB RAS [6]. Ethernet-порты интерфейсных плат DWDM-тракта на участке Пермь – Екатеринбург имеют скорости передачи 1–10 Гбит/с.

Расчетное время распространения сигнала для трассы Пермь – Екатеринбург с оптической длиной 456 км равно 2,28 мс. Эта величина задержки экспериментально подтверждается измерением RTT (Round Trip Time) между пограничными коммутаторами в Перми и Екатеринбурге, которая с учетом задержек в коммутационном облаке (L2 OSI RM) будет

rtt min/avg/max/mdev = 5.285/5.300/5.341/0.102 мс (канал связи 1 Гбит/с);

rtt min/avg/max/mdev = 5.198/5.210/5.237/0.077 мс (канал связи 10 Гбит/с).

Параметр RTT для тестовой петли Пермь – Екатеринбург – Пермь с пропускной способностью 10 Гбит/ будет

rtt min/avg/max/mdev = 10.491/10.604/11.073/0.129 ms

Кластер ИМСС является MPP-системой с Infiniband- и Ethernet-интерконнектом и пиковой производительностью 4,5 Tflops. Кластер состоит из трех базовых блоков HP BladeSystem c7000, в каждом из которых находится 16 серверов HP Proliant BL 460c. Вычислительные узлы кластера для работы используют две сети передачи данных. Основной (счетной) сетью обмена данными MPI является InfiniBand 4xDDR пропускной способностью 20 Гбит/с. Дополнительная сеть Ethernet, пропускной способностью 1 Гбит/с, предназначена для управления потоком задач и монтирования файловых систем на вычислительные узлы.

Каждый базовый блок оборудован встроенным Ethernet-коммутатором, к которому на скорости 1 Гбит/с подключен каждый вычислительный узел. Коммутаторы базовых блоков подключены к внешнему коммутатору AS9215 при помощи четырех агрегированных по технологии LACP соединений 1 Гбит/с (суммарная пропускная способность агрегированного канала 4 Гбит/с). Коммутатор AS9215 подключен к DWDM-тракту Пермь – Екатеринбург на скорости 10 Гбит/с.

Менеджер/диспетчер [2, 7] установлен на отдельном сервере очередей HP ProLiant DL360p Gen8 (2x Intel Xeon CPU E5-2660, 2,20 ГГц, 16 потоков; RAM 128 Гб; операционная система CentOS 7.0, сетевая карта 10GE Mellanox Technologies MT25400 [ConnectX-2]).

Размер окна для используемых в SciMP транспортных протоколов с обратной связью определяется посредством вычисления $BDP = bandwidth*RTT$, которая равна 0,6625 или 6,5125 Мбайт для скоростей 1 Гбит/с (RTT = 5,3 мс) и 10 Гбит/с (RTT = 5,21 мс) соответственно при длине линии связи 456 км. При увеличении длины линии связи с 456 до 900 км параметр BDP следует увеличить в два раза.

Для эффективной работы алгоритмов управления перегрузкой в TCP требуется, чтобы размер окна перегрузки cwnd был не меньше BDP.

Измерения

Для сравнения зависимости пропускной способности разработанного сервера очередей SciMQ и существующего сервера очередей RabbitMQ от размера сообщения и количества параллельных запросов к серверу были проведены измерения. Программное обеспечение RabbitMQ и SciMQ было развернуто на первом сервере HP ProLiant DL360p Gen8, а источник и приемник сообщений – на втором сервере, которые были соединены каналом связи 10 Гбит/с через коммутатор ECI AS9215. Количество сообщений, передаваемых в рамках одного теста, подбиралось таким образом, чтобы помещаться в оперативную память серверов RabbitMQ и SciMQ.

Из графиков (рис. 7) следует, что при использовании программного обеспечения SciMQ на требуемых на практике размерах сообщений (2–22 Мбайт) пропускная способность ограничивается в основном доступной скоростью канала связи.

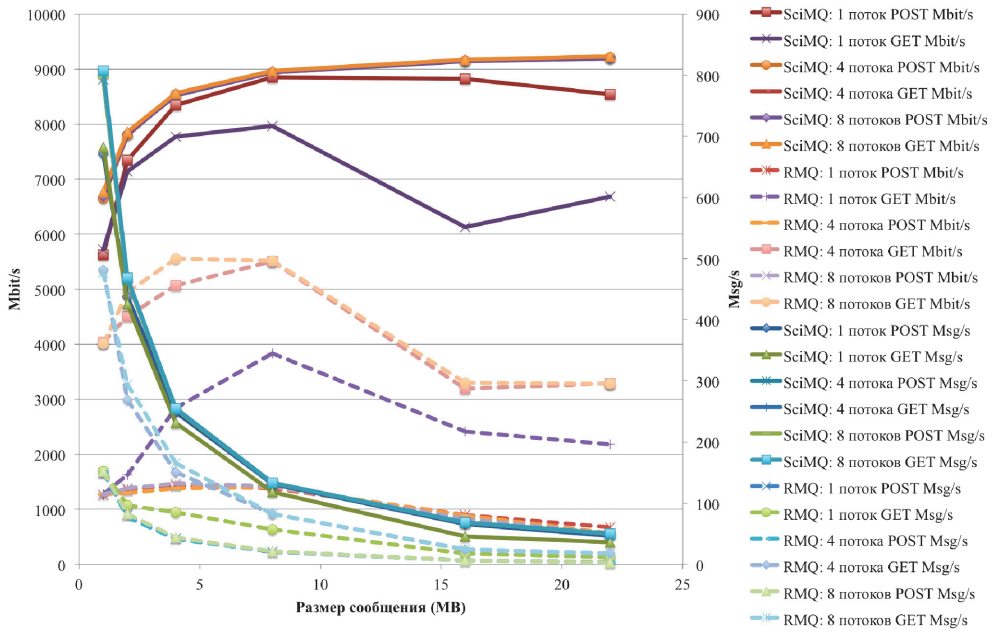

Рис. 7. Сравнение производительности разработанного сервера очередей SciMQ с существующим RabbitMQ (RMQ)

Сравнение полученных результатов с результатами тестирования производительности возможных готовых решений на примере RabbitMQ показывает, что разработанное программное обеспечение SciMQ по-

зволяет диспетчеризировать сообщения размером более 1 Мбайта в 2–4 раза быстрее, чем в RabbitMQ.

Для изучения стабильности работы системы во времени было проведено тестирование в следующей конфигурации (см. рис. 6): сервер очередей был подключен на скорости 10 Гбит/с к внутреннему коммутатору интерконнекта кластера ИМСС УрО РАН через тестовую петлю Пермь – Екатеринбург – Пермь (900 км). Поток исходных данных состоял из сообщений размером 16 Мбайт, которые передавались каждые 20 мс, порождая трафик со скоростью 6,4 Гбит/с. В эксперименте было передано 1800 000 сообщений общим объемом 27,5 Тбайт. За получение данных отвечали 376 процессов, запущенных на кластере ИМСС УрО РАН.

Результаты этого эксперимента, показанные на рис. 8, имеют следующие характерные показатели работы сервера / менеджера очередей значения: среднее число сообщений в обработке – 325, среднее число параллельно передаваемых сообщений – 90. Среднее значение интервала генерации сообщений (POST) равно среднему интервалу запросов от вычислительных узлов (GET), что является условием устойчивой работы тракта передачи данных в целом.

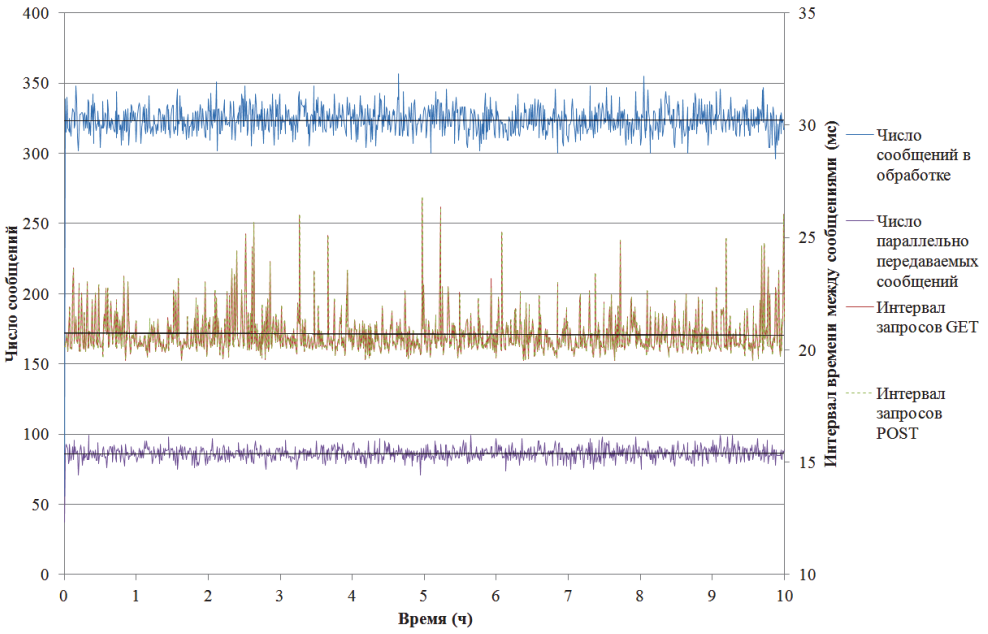

Рис. 8. Графики изменения числа сообщений в обработке, числа параллельно передаваемых сообщений и средних интервалов между поступлением исходных сообщений и отправкой сообщений клиентам

Пиковое потребление оперативной памяти сервером очередей равно 5,46 Гбайт (350 сообщений по 16 Мбайт), а среднее – 5,07 Гбайт (325 сообщении по 16 Мбайт), что совпадает со статистикой потребления ресурсов, которую ведет программное обеспечение.

Из приведенных графиков следует:

– 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 способно передавать продолжительные по времени интенсивные потоки данных.

Макроанализ результатов измерений

Детальный анализ данных эксперимента позволил определить несколько факторов, отрицательно влияющих на производительность системы.

Первым фактором является несимметричность подключения серверов к сети передачи данных. Как следует из схемы тестовой инфраструктуры, приведенной на рис. 6, сервер очередей подключен к сети передачи данных на скорости 10 Гбит/с, а вычислительные узлы – 1 Гбит/с. При этом каждая из трех групп по 16 вычислительных узлов подключена к пограничному коммутатору кластера ИМСС по четырем каналам пропускной способностью 1 Гбит/с, агрегированных по технологии LACP [8].

Технология LACP обеспечивает распределение входящего и исходящего трафика по каналам связи на основе адресов отправителя и получателя. В нашем случае используется балансировка на основе MAC-адресов.

Такой механизм распределения трафика по каналам приводит к тому, что максимальная пропускная способность может быть достигнута только при одновременной передаче данных всеми 16 вычислительными узлами группы. В случае передачи данных меньшему числу вычислительных узлов может оказаться так, что потоки трафика будут направлены только в часть доступных каналов, в то время как остальные будут простаивать.

Всё это приводит к тому, что максимальная пропускная способность Ethernet-интерконнекта кластера ИМСС реально зависит от параметров потоков данных и от расположения используемых вычислительных узлов в коммуникационной среде и может составлять от 3 до 12 Гбит/с.

Вторым фактором, вызывающим снижение пропускной способности системы, является совместная работа алгоритмов управления потоком TCP и механизмов аппаратной разгрузки процессора, встроенных в сетевую карту сервера очередей.

Технологии scatter-gather и tcp-segmentation-offload [9, 10] позволяют существенно снизить загрузку процессора при симметричной конфигурации сети (и отправитель, и получатель подключены на скорости 10 Гбит/с), но негативно влияют на работу алгоритмов управления потоком TCP в асимметричной конфигурации сети. Суть механизма tcp-segmentation-offload заключается в передаче задачи сегментации потока данных сетевой карте. Обычно блок, передаваемый сетевой карте, равен 64 Кбайт. Это приводит к тому, что после сегментации все полученные пакеты будут отправлены получателю на скорости среды передачи данных (10 Гбит/с). Если получатель подключен на такой же скорости, то все отправленные сегменты будут получены. Если же получатель подключен на меньшей скорости, например 1 Гбит/с, то он сможет получить в лучшем случае только каждый десятый сегмент, в то время как остальные будут отброшены коммутатором. Это приведет к резкому росту потерь и, как следствие, снижению темпа отправки данных алгоритмами управления потоком в TCP. Поскольку при последующих отправках механизмы scatter-gather и tcp-segmentation-offload продолжат работу, то ситуация повторится.

Эксперименты на тестовой петле Пермь – Екатеринбург – Пермь показали, что снижение скорости может достигать 4 раз при передаче данных на управляющий узел кластера ИМСС и до 10 раз при передаче данных на отдельный вычислительный узел (которые подключены с использованием промежуточной LACP-агрегации) в зависимости от настроек scatter-gather и tcp-segmentation-offload.

Заключение

Разработанные архитектурные решения предоставляют принципиально новый инструмент проведения уникальных физических экспериментов в исследовательских лабораториях и на производстве. Отличительной особенностью создаваемой технологии является асинхронная модель передачи потока данных от источника в удаленный суперкомпьютер, основанная на идее параллельного прямого ввода структурированного потока данных в вычислительные узлы суперком-

пьютера. 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. Начато внедрение разработанных технологий в проекте по созданию технологической платформы для экспериментальных и вычислительных исследований быстропротекающих процессов гидроупругости на базе связки измерительной системы с суперЭВМ.

Исследование проводится при поддержке РФФИ (гранты № 14-07-96001 и №14-07-96003).

Библиографический список

1. Обработка на супервычислителе потока экспериментальных данных / Р.А. Степанов, А.Г. Масич, А.Н. Сухановский, В.А. Щапов, А.С. Игумнов, Г.Ф. Масич // Вестник УГАТУ. – Уфа, 2012. – Т. 16, № 3 (48). – С. 126–133.

2. Щапов В.А., Масич А.Г., Масич Г.Ф. Модель потоковой обработки экспериментальных данных в распределенных системах // Вычислительные методы и программирование. – 2012. – Разд. 2. – С. 139–145.

3. AWS | Amazon Simple Queue Service (SQS) – Queue Messaging Service, available at: http://aws.amazon.com/sqs (дата обращения: 23.10.2014).

4. Vinoski S. Advanced Message Queuing Protocol // Internet Computing, IEEE. – 2006. – Vol. 10, no. 6. – P. 87–89.

5. The Intelligent Transport Layer – zeromq, available at: http://www.zeromq.org (дата обращения: 23.09.2014).

6. Masich A.G., Masich G.F. Initiative GIGA UrB RAS // Computational Technologies. – Vol. 13. The Bulletin of Kaznu. Mathematics, Mechanics, and Informatics Issue. – № 3(58). – Almaty (Kazakhstan) – Novosibirsk (Russia), 2008. – Part II. – P. 413–418 [in Russian].

7. Shchapov V., Masich A. 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. Link Aggregation Control Protocol, available at: https://ru.wikipedia.org/wiki/Link_Aggregation_Control_Protocol (дата обращения: 23.09.2014).

9. gso | The Linux Foundation, available at: http://www.linuxfoundation.org/collaborate/workgroups/networking/gso (дата обращения: 23.09.2014).

10. tso | The Linux Foundation, available at: http://www.linuxfoundation.org/collaborate/workgroups/networking/tso (дата обращения: 23.09.2014).

References

1. Stepanov R.A., Masich A.G., Sukhanovskii A.N., Shchapov V.A., Igumnov A.S., Masich G.F. Obrabotka na supervychislitele potoka eksperimentalnykh dannykh [Experimental data stream processing on supercomputer]. *Vestnik Ufimskogo gosudarstvennogo aviatsionnogo tekhnicheskogo universiteta*, 2012, vol. 16, no. 3 (48), pp. 126-133.

2. Shchapov V.A., Masich A.G., Masich G.F. Model potokovoy obrabotki eksperimentalnykh dannykh v raspredelennykh sistemakh [A model of stream processing of experimental data in distributed systems]. *Vychislitelnye Metody i Programmirovanie*, 2012, section 2, pp. 139-145.

3. AWS | Amazon Simple Queue Service (SQS) – Queue Messaging Service, available at: http://aws.amazon.com/sqs (accessed 23 September 2014).

4. Vinoski S. Advanced Message Queuing Protocol. *Internet Computing, IEEE*, 2006, vol. 10, no. 6, pp. 87-89.

5. The Intelligent Transport Layer – zeromq, available at: http://www.zeromq.org (accessed 23 September 2014).

6. Masich A.G. Masich G.F. Initiative GIGA UrB RAS. *Computational Technologies, vol. 13. The Bulletin of Kaznu. Mathematics, Mechanics, and Informatics Issue*, no. 3(58), Almaty (Kazakhstan), Novosibirsk (Russia), 2008, part II, pp. 413-418 [in Russian].

7. Shchapov V., Masich A. 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, 2012, vol. 2, pp. 653-657. DOI: 10.1109/IFOST.2012.6357642

8. Link Aggregation Control Protocol, available at: https://ru.wikipedia.org/wiki/Link_Aggregation_Control_Protocol (accessed 23 September 2014).

9. gso | The Linux Foundation, available at: http://www.linuxfoundation.org/collaborate/workgroups/networking/gso (accessed 23 September 2014).

10. tso | The Linux Foundation, available at: http://www.linuxfoundation.org/collaborate/workgroups/networking/tso (accessed 23 September 2014).

Сведения об авторах

Масич Григорий Федорович (Пермь, Россия) – кандидат технических наук, старший научный сотрудник, заведующий лабораторией «Телекоммуникационные и информационные системы» Института механики сплошных сред УрО РАН (614013, г. Пермь, ул. Академика Королева, 1, e-mail: masich@icmm.ru).

Щапов Владислав Алексеевич (Пермь, Россия) – младший научный сотрудник лаборатории «Телекоммуникационные и информационные системы» Института механики сплошных сред УрО РАН (614013, г. Пермь, ул. Академика Королева, 1, e-mail: shchapov@icmm.ru).

About the authors

Grigoriy F. Masich (Perm, Russian Federation) – Ph. D. in Technical Sciences, Senior Research Fellow of Institute of Continuous Media Mechanics, Ural Branch of Russian Akademy of Sciences (1, Akademika Koroleva st., Perm, 614013, Russian Federation, e-mail: masich@icmm.ru)

Shchapov A. Vladislav (Perm, Russian Federation) – Junior Research Fellow of Institute of Continuous Media Mechanics, Ural Branch of Russian Akademy of Sciences (1, Akademika Koroleva st., Perm, 614013, Russian Federation, e-mail: shchapov@icmm.ru).

Получено 1.10.2014