Экспериментальная оценка эффективности модели очередей и алгоритмов параллельной передачи в задачах обработки интенсивных потоков данных в распределенных системах*

В.А. Щапов
Институт механики сплошных сред УрО РАН
Пермский национальный исследовательский политехнический университет
e-mail: shchapov@icmm.ru

А.Г. Масич
Институт механики сплошных сред УрО РАН
e-mail: mag@icmm.ru

Г.Ф. Масич
e-mail: masich@icmm.ru

В работе рассмотрена разработанная модель очередей, используемая для параллельной передачи данных по скоростной протяженной линии связи. Приведен алгоритм распределения параллельных потоков данных от источника по вычислительным узлам удаленной суперЭВМ и его зависимость от стратегий управления перегрузками в транспортных протоколах. Описана реализация модели в виде специализированного комплекса программного обеспечения, устанавливаемого во взаимодействующих оконечных системах и включающего протокол обмена сообщениями между прикладными процессами вычислительных узлов суперЭВМ и источником интенсивного потока данных. Показана возможность динамической адаптации числа вычислительных узлов к скорости входного потока и времени его обработки прикладными программами, в том числе на нескольких суперкомпьютерах. Приводятся результаты измерений и эмпирической оценки эффективности модели очередей и алгоритмов управления ими при передаче интенсивного потока данных по высокоскоростной оптической магистрали Пермь-Екатеринбург. Показана область применения, и эффективность разработанных решений для обработки потока экспериментальных данных на суперЭВМ в реальном времени.

1. Введение

Тенденции настоящего времени показывают экспоненциальный рост объемов данных, которые генерируются во всех сферах человеческой деятельности, таких как наука, промышленность, экономика и т.д. Соответственно, растут требования и к производительности вычислительных систем, необходимых для их обработки и хранения. В этих условиях особенно актуальной становится проблема передачи интенсивных потоков данных от мест их генерации к местам их обработки и хранения [1]. Для достижения практических результатов эта проблема должна решаться комплексно, на всех уровнях. Аппаратным фундаментом в данном случае являются параллельные скоростные сети,

*Работа выполнена при поддержке РФФИ (грант №11-07-96001-р_урал_а).

примером которых может служить создаваемая в рамках проекта «Инициатива GIGA UrB RAS» [2] научно образовательная сеть УрО РАН.

Второй компонентой являются модели передачи данных и алгоритмы их диспетчеризации по вычислителям, а так же промежуточное программное обеспечение, решающее эти задачи. Для случая, когда исходный поток данных допускает разбиение на сообщения, которые можно независимо обрабатывать или сохранять, нами была предложена модель очередей, использующая лямбда-грид парадигму распределенных вычислений [3], в которой используется параллелизм потоков данных в скоростных оптических сетях со спектральным уплотнением каналов. Достижение эффективной диспетчеризации параллельных потоков сообщений между сопрягаемыми системами является основной целью представленной работы. Апробация разработанных технологий проводилась с использованием выделенного L2 канала связи, который объединяет в единую подсеть источник данных (Пермь, ИМСС УрО РАН), суперкомпьютеры «Уран» и «UM64» и систему хранения данных (СХД) EMC Celerra NS-480 (Екатеринбург, ИММ УрО РАН). В настоящий момент доступная пропускная способность выделенного канала связи со-ставляет 1 Гбит/с.

2. Трехуровневая архитектура системы обработки потока данных

Концепция очередей данных, показанная на рисунке 1, для решения задачи передачи и диспетчеризации потока данных напрямую вытекает из возможности разбиения исходного потока данных на подмножества (сообщения), допускающие полностью независимую обработку [4].

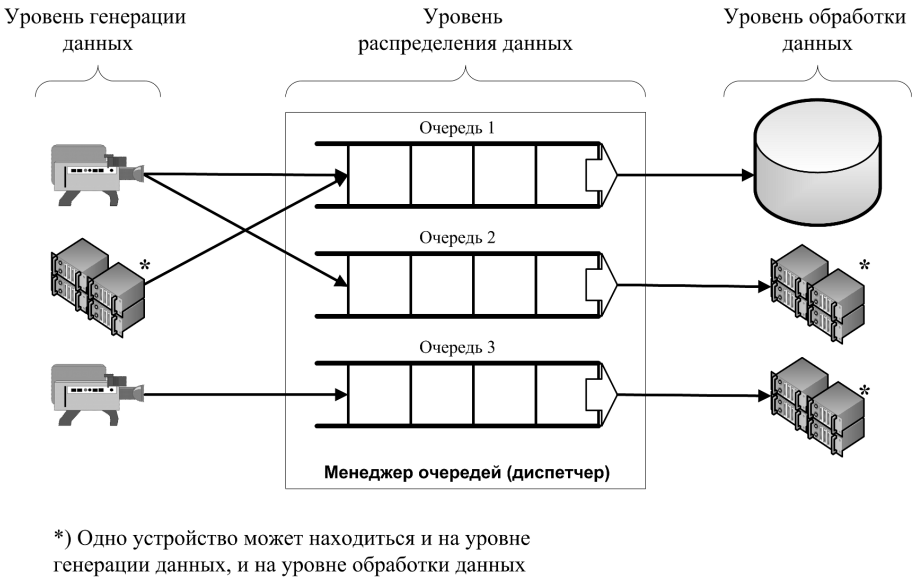

Рис. 1. Трехуровневая архитектура системы обработки потока данных

Уровень генерации данных отвечает за загрузку исходных данных из источника данных и отправку их на уровень распределения в виде потока сообщений, допускающих независимую обработку.

Уровень распределения данных решает задачу получения сообщений от уровня генерации данных и их передачу на уровень обработки данных. В связи с тем, что уровень обработки данных состоит из множества вычислительных процессов, запущенных на узлах суперкомпьютера, на диспетчер ложится задача по распределению очередей данных на множество узлов суперкомпьютера. Исходя из независимости и равнозначности передаваемых сообщений, было принято решение использовать First In First Out (FIFO) стратегию распределения данных по запросам от оконечных систем.

Уровень обработки данных отвечает за обработку данных в терминах прикладной задачи. Это может быть расчет исходных данных с применением каких-либо алгоритмов, сохранение данных в высокопроизводительных хранилищах или на большом количестве локальных дисков, передача данных из очередей в сторонние системы и т.д.

Оконечные системы взаимодействуют с менеджером очередей при помощи прикладного протокола, получившего название «Протокол PIV» (рисунок 2). Для прикладных программистов протокол реализован в виде клиентской библиотеки, написанной на языке C++, которую могут использовать приложения уровня генерации и обработки данных.

Рис. 2. Схема взаимодействия компонент

Выделение задачи диспетчеризации в отдельный слой позволило отказаться от межузлового обмена данными на стороне суперкомпьютера, что дает следующие преимущества:

- автоматическая балансировка нагрузки по вычислительным узлам – более быстрые узлы будут чаще посылать запросы новых данных и, таким образом, будут получать больше данных для обработки;
- возможность изменять число используемых вычислительных узлов во время проведения эксперимента;
- возможность использовать вычислительную мощность нескольких суперкомпьютеров;
- минимизация потери данных в случае выхода из строя одного или нескольких вычислительных узлов (в худшем случае потеряется только текущее обрабатываемое измерение сбойного узла).

Перенос менеджера очередей на отдельный сервер позволяет снизить нагрузку на источник данных, так как в этом случае задача по обслуживанию большого числа параллельных TCP-соединений решается выделенным сервером, не используя ресурсы источника данных. Установка менеджера очередей недалеко от источника данных позволит эффективно передавать данные диспетчеру через небольшое число соединений.

3. Протокол PIV

Протокол PIV является протоколом прикладного уровня, реализующим идею модели RPC. Протокол работает по схеме запрос-ответ и может использовать в качестве протокола транспортного уровня любой надежный потоковый протокол передачи данных. Текущая реализация протокола PIV поддерживает транспортные протоколы TCP и UDT [5]. Положение протокола PIV в стеке сетевых протоколов показано на рисунке 3 [6].

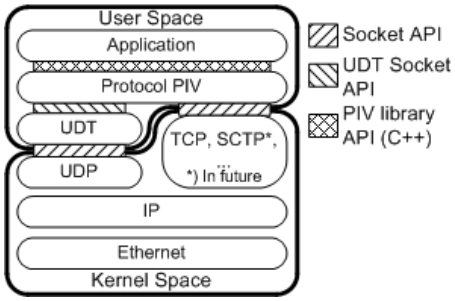

Рис. 3. Положение протокола PIV в стеке сетевых протоколов

Протокол PIV рассчитан на передачу нескольких именованных блоков бинарных данных. В одном пакете протокола PIV можно передать от нуля до 65535 блоков, каждый из которых может иметь размер до 4 Гбайт, при этом сохранение порядка следования блоков не гарантируется. Длина имени блока ограничена 255 байтами. На рисунке 4 приведен формат пакета протокола PIV. Поля заголовка пакета протокола кодируются в сетевом порядке байт.

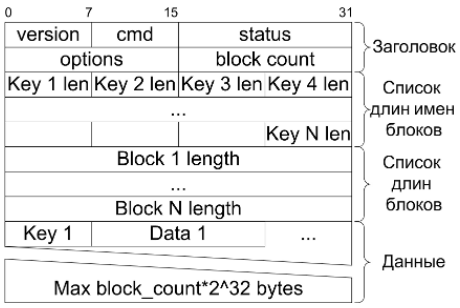

Рис. 4. Формат пакета протокола PIV

Протокол поддерживает версионирование с использованием поля `version`, что позволяет реализовать в рамках одной версии ПО поддержку нескольких версий протокола. Поле `cmd` содержит код выполняемой команды. Поле `status` предназначено для передачи клиенту статуса ответа от сервера. В зависимости от значения статуса можно судить о выполнении или невыполнении запроса клиента. Поле `options` предназначено для кодирования дополнительных опций. Поле `block_count` содержит число передаваемых блоков данных. Далее в пакете располагаются список длин имен блоков и список длин блоков, после чего передаются пары, состоящие из имени блока и данных блока.

На рисунке 5 приведена временная диаграмма работы протокола PIV в трехуровневой архитектуре. Левая часть диаграммы показывает взаимодействие уровня генерации

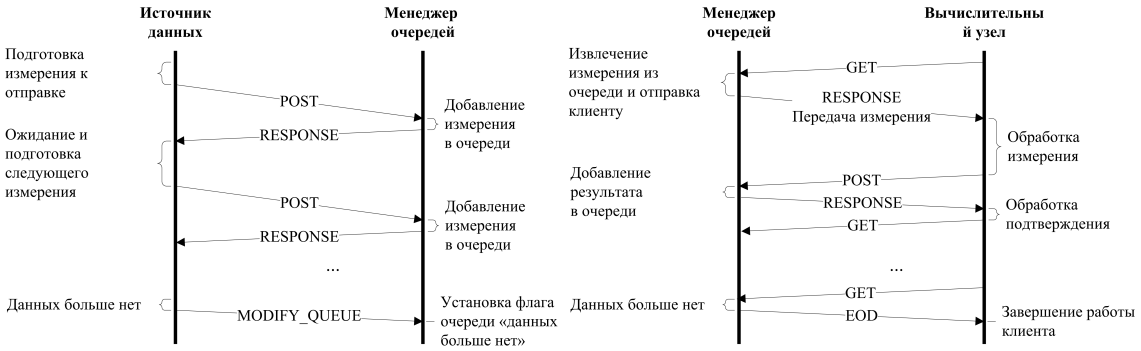

Рис. 5. Временная диаграмма работы протокола PIV

данных, а правая часть – уровня обработки данных с менеджером очередей. Для наглядности эти процессы разделены, однако существенным является то обстоятельство, что передача по протяженной линии связи занимает значительное время и процессы передачи необходимо выполнять одновременно при наличии данных.

4. Программная реализация

Предложенная модель была реализована в виде комплекса ПО для всех уровней системы. Были разработаны три компоненты общего назначения:

- сервер очередей,
- клиентская библиотека,
- управляющее ПО.

Все приложения и библиотеки написаны на языке программирования C++ с использованием библиотек Boost[1]. В связи с тем, что менеджер очередей не занимается сложной обработкой поступающих данных, его работа с сетевыми сокетами реализована по событийно-ориентированному принципу с использованием технологии асинхронных, неблокирующих сетевых операций. Так как в разных операционных системах сетевые API значительно отличаются, для унификации кода используется библиотека Boost.ASIO, которая предоставляет единый интерфейс для различных технологий асинхронного I/O в различных операционных системах.

5. Оценка эффективности разработанного решения

Оценка эффективности проводилась путем измерения эффективной пропускной способности системы при передаче данных от экспериментальной установки через сервер очередей на площадке ИМСС УрО РАН (Пермь) на вычислительные узлы удаленного суперкомпьютера «Уран» ИММ УрО РАН (Екатеринбург). Связь между городами осуществлялась по выделенному vlan по DWDM магистрали «GIGA UrB RAS» с доступной пропускной способностью 1 Гбит/с и временем RTT=5,5 мс.

На первом этапе оценивалась эффективность существующих в операционной системе Linux алгоритмов контроля перегрузки TCP. Для этого были проведены тесты с

[1]Boost C++ Libraries: http://www.boost.org/

использованием ПО iperf. В рамках тестирования для каждого из имеющихся алгоритмов iperf запускался на 60 секунд с использованием 1, 4, 16, 64 параллельных потоков TCP. Результаты измерений показаны на рисунке 6

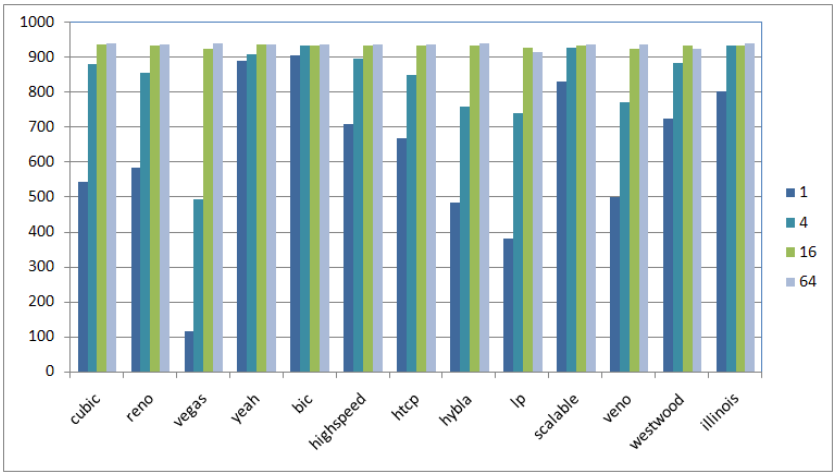

Рис. 6. Оценка эффективности различных реализаций алгоритмов управления перегрузкой TCP

Из графика следует, что используемый по умолчанию в Linux алгоритм cubic при малом числе параллельных потоков показывает явно недостаточную эффективность. Алгоритмы yeah, bic, highspeed, scalable и illinois показали большую эффективность утилизации канала связи, однако не все из них доступны в промышленных дистрибутивах Linux, а так же для изменения используемого алгоритма требуются права администратора, что невозможно в случае использования сторонних суперкомпьютеров. Необходимо отметить, что в используемых версиях Windows в настоящее время эффективные алгоритмы управления перегрузкой отсутствуют.

На втором этапе исследовалось поведение системы при передаче сообщений, состоящих из трех блоков с длинами 128, 1048576, 1048576 байт, соответственно. Измерения поступали от эмулятора экспериментальной установки PIV (ЭУ) с промежутками между отправками данных, равными 20 мс, 50 мс и 100 мс. Это приближенно соответствует нескольким режимам работы реальной ЭУ. При этом при минимальном интервале скорость генерации составляет порядка 750 Мбит/с, что меньше предельной пропускной способности канала связи и поэтому позволяет работать в реальном времени без бесконечного роста размера очереди исходных данных. Процесс на вычислительных узлах рассчитывал контрольные суммы блоков данных по алгоритму CRC32 и передавал их обратно на эмулятор ЭУ.

Менеджер очередей располагался на Linux-сервере, который был настроен на использование алгоритма управления перегрузкой illinois.

На рисунке 7 показаны графики зависимости пропускной способности системы от числа задействованных вычислительных узлов. Для наглядности горизонтальная ось «число вычислительных узлов» приведена в логарифмическом масштабе.

Наклонные участки показывают рост пропускной способности системы с добавлением новых узлов, когда меньшее количество узлов не справляются с обработкой и небольшое число параллельных соединений не полностью утилизируют канал связи. При работе в этой области графика очередь растет, так как вычислительные узлы не успевают обрабатывать все данные в реальном времени.

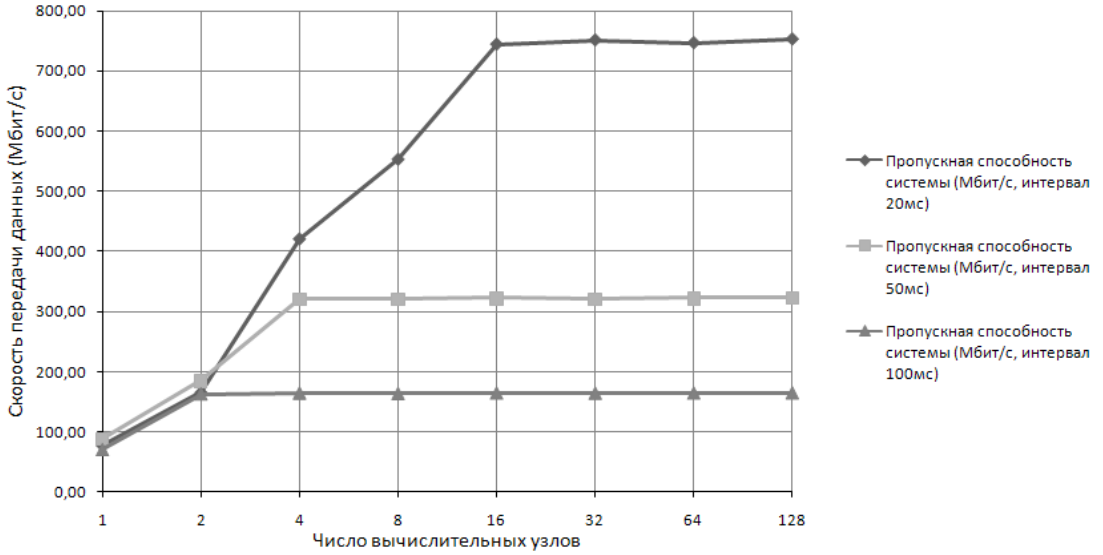

Рис. 7. Зависимость пропускной способности системы от числа задействованных вычислительных узлов

Горизонтальные участки в правой части графика соответствуют установившемуся режиму, в котором скорость обработки данных совпадает со скоростью генерации данных. В этом случае увеличение числа задействованных вычислительных узлов не влияет на пропускную способность системы, но увеличивает ее надежность. Рост надежности обусловлен тем, что в случае отключения или потери сетевой связности части вычислительных узлов, а также в случае, если время обработки одного измерения по каким-либо причинам возрастет, то система все равно может остаться в рамках горизонтального участка графика и не допустить снижение пропускной способности ниже скорости генерации данных. Необходимо отметить, что разработанная архитектура позволяет, при необходимости, увеличивать число задействованных вычислительных узлов непосредственно в процессе работы системы, что предоставляет возможность оперативно реагировать на изменения условий проведения расчета.

6. Заключение

Предложена модель очередей для параллельной передачи данных по скоростной протяженной линии связи в распределенных системах.

Приведен алгоритм распределения параллельных потоков данных от источника по вычислительным узлам удаленной суперЭВМ и его зависимость от стратегий управления перегрузками в транспортных протоколах. Спроектирован протокол, алгоритм диспетчеризации данных и разработано программное обеспечение для передачи данных с экспериментальной установки на узлы вычислительного кластера.

Проведенное экспериментальное сравнение реализаций алгоритмов управления перегрузкой TCP показало, что различные алгоритмы могут значительно отличаться по эффективности и что параметры по умолчанию в настройках операционных систем могут быть не оптимальными в конкретных ситуациях. В связи с тем, что изменение параметров сетевых подсистем требует администраторских полномочий и не всегда возможно, была обоснована необходимость создания специализированных решений, ко-

торые могут повышать суммарную эффективность системы в случае, если параметры сетевого стека не удовлетворяют необходимым требованиям.

Оценка эффективности модели очередей и протокола PIV показала увеличение пропускной способности системы при использовании разработанной технологии по сравнению с классическими подходами [7] и, как следствие, увеличение предельной пропускной способности системы. Показан возможный путь снижения нагрузки на процессор ЭУ путем переноса задачи диспетчеризации на близкорасположенный выделенный сервер. Помимо этого, использование выделенного сервера позволяет проводить сборку набора данных из нескольких несвязанных между собой источников, например, при наличии нескольких независимых групп датчиков, допускающих раздельную обработку данных с них.

Разработанная технология применяется в ИМСС УрО РАН в рамках проекта «Распределенный PIV» [7] для обработки экспериментальных данных, получаемых по методу PIV (Particle Image Velocimetry) – оптическому методу измерения полей скорости жидкости или газа в выбранном сечении потока, на удаленном суперкомпьютере в реальном времени.

Список литературы

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

[2] А.Г. Масич, Г.Ф. Масич Инициатива GIGA UrB RAS // Совместный вып. Журнала «Вычислительные технологии» и журн. «Вестник КазНУ им. Аль-Фараби». Сер. «Математика, механика, информатика», №3 (58). По материалам Междунар. конф. «Вычислительные и информационные технологии в науке, технике и образовании», - Казахстан, Алматы.-2008.-Т.13.- Ч. II. -С. 413-418 (ISSN 1560-7534).

[3] А.Г. Масич, Г.Ф. Масич GIGA UrB RAS подход к LambdaGrid парадигмам вычислений // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: Труды Международной суперкомпьютерной конференции (20-25 сентября 2010 г., г. Новороссийск). – М.: Изд-во МГУ, 2010. С. 4-11.

[4] А.Г. Масич, Г.Ф. Масич, Р.А. Степанов, В.А. Щапов Скоростной I/O-канал супервычислителя и протокол обмена интенсивным потоком экспериментальных данных // Материалы X международной конференции «Высокопроизводительные параллельные вычисления на кластерных системах HPC-2010» – Пермь: Изд-во ПГТУ, 2010. - Т. 2. С. 119–128. (ISBN 978-5-398-00506-6).

[5] Yunhong Gu and Robert L. Grossman UDT: UDP-based Data Transfer for High-Speed Wide Area Networks, Computer Networks (Elsevier). Volume 51, Issue 7. May 2007.

[6] 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, 2012, Vol.1. National Research Tomsk Polytechnic University, Tomsk, P. 653-657

[7] Р.А. Степанов, А.Г. Масич, Г.Ф. Масич Инициативный проект «Распределенный PIV» // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: труды Всероссийской суперкомпьютерной конференции – М.: Изд-во МГУ, 2009. – С. 360–363. (ISBN 978-5-211-05697-8).