Анализ эффективности масштабируемых подходов к решению задач с преобладанием ввода-вывода*

Д.Ю. Андреев[1,2], О.В. Джосан[1]

МГУ имени М.В.Ломоносова[1], ВЦ ДВО РАН имени А.А. Дородницина[2]

В данной работе описываются подходы к организации ввода-вывода в задачах с потоком входных - выходных данных и высокой загрузкой каналов, соединяющих вычислительные узлы и внешние системы хранения данных. Разработаны методы разделения функциональности вычислительных узлов в рамках модели иерархии коммуникаторов. Авторами предлагается библиотека, реализующая описанные методы. Исследование производительности разработанной библиотеки проводилось на системах BlueGene /P и «Ломоносов».

1. Введение

В настоящее время в мире решается множество задач по обработке данных, параллельные алгоритмы создаются и используются на различных архитектурах вычислительных систем. При реализации таких алгоритмов перед программистом встает вопрос об оптимальном способе загрузки данных из внешнего хранилища информации, распределении данных между узлами вычислительной системы.

Современные вычислительные системы содержат множество процессоров, которые могут одновременно обращаться за данными во внешнем хранилище. Увеличение числа вычислительных узлов приводит к усложнению архитектуры коммуникационной среды и систем хранения данных. При неэффективном использовании коммуникационных соединений или при неудачном сочетании обращений к жестким дискам во внешнем хранилище производительность загрузки данных может ухудшаться из-за ограниченности пропускной способности сети или скорости работы жестких дисков. Прикладному программисту сложно разрабатывать универсальные алгоритмы для всего разнообразия архитектур вычислительных систем.

2. Основные идеи и понятия

Не во всех задачах загрузка входных данных и выгрузка результатов занимает значительную часть времени работы. Проблема производительности операций ввода-вывода ярко выражена в задачах обработки потока данных большого размера, например при обработке последовательности изображений. В таких задачах часто можно встретить использование данных представимых в виде многомерных массивов. Существует не одно решение вопроса о формате хранения многомерных массивов, но наиболее популярными и признанными в научном мире являются библиотеки ориентированные на бинарный формат хранения netCDF[1] и HDF[2]. Многие современные архитектурные решения многопроцессорных вычислительных систем [8] предоставляют хранилище данных с возможностью параллельного доступа к файлам (с параллельными файловыми системами GPFS, PVFS, Lustre, и т. д.), существуют версии библиотек netCDF и HDF позволяющие использовать параллельный доступ к частям многомерных массивов. В концепции разрабатываемой библиотеке определено абстрактное понятие «источник данных». Указанные выше файлы на внешнем хранилище в форматах netCDF, HDF можно использовать в качестве источника данных.

В большинстве архитектур многопроцессорных вычислительных систем доступ к внешнему хранилищу неоднороден. Это обычно связанно с тем, что на несколько вычислительных узлов приходится один узел ввода-вывода. Например, в конфигурации системы BlueGene /P[3], установленной на факультете вычислительной математики и кибернетики МГУ имени М.В.

* Работа ведтся при поддержке Федеральной целевой программы "Научные и научно-педагогические кадры инновационной России" на 2009 — 2013гг.

Ломоносова, 128 вычислительных узлов используют один узел ввода-вывода. Неравномерное распределение между каналами ввода-вывода приводит к понижению производительности. В статье [4] демонстрируется, что в вычислительных системах с подобной организацией подсистемы ввода-вывода эффективным решением является выделение вычислительных узлов, через которые будут проходить все операции ввода-вывода. Естественным видится, что выделенные вычислительные узлы должны использовать максимальное доступное число каналов ввода-вывода. Вышеописанное приводит ко второму абстрактному понятию — «I/O вычислительные узлы».

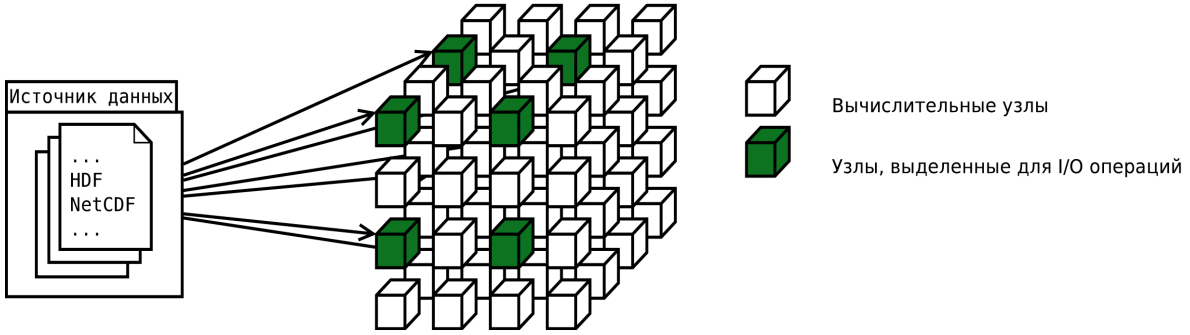

Рис. 1. Загрузка данных на выделенные узлы

Загрузив данные на выделенные вычислительные узлы, необходимо распределить их требуемым образом по вычислительным узлам обработки данных. В случае обработки потока данных, часто загрузка/выгрузка данных является более времязатратной операцией, чем обработка. Эффективным решением является продолжать загрузку/выгрузку данных одновременно с обработкой на вычислительных узлах, соответственно для обработки данных используется узлы не участвующие в операциях ввода-вывода. Назовём подмножество вычислительных узлов, обрабатывающих данные, областью обработки. Таким образом, в одной программе может быть несколько областей обработки, производящих разную работу над данными.

Схема работы программы в описанных терминах:

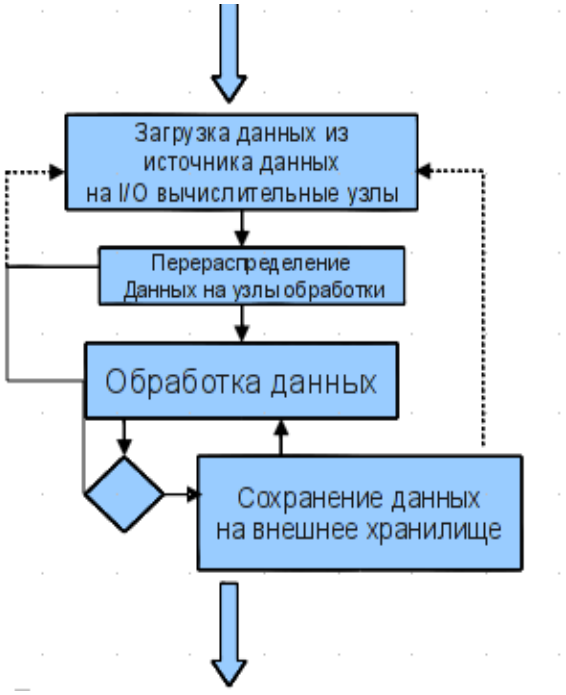

Рис. 2. Схема работы программы

При разработке библиотеки, реализующей описанные выше концепции, одной из целей было предусмотреть удобную возможность дальнейшей доработки. Части библиотеки, отвечающие разным этапам программы, связанны с друг другом только через интерфейс соответствующих классов или параметры функций. Внутренняя реализация библиотеки разрабатывается на языке C++ и имеет интерфейсы на языках C и C++.

В качестве библиотеки для организации многопроцессорного режима и обмена данными между вычислительными узлами используются самая популярная в научной среде библиотека Message Passing Interface. Message Passing Interface (MPI, интерфейс передачи сообщений) — программный интерфейс для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. В первую очередь MPI ориентирован на системы с распределенной памятью (большинство суперкомпьютеров являются системами с распределённой памятью).

3. Возможности разрабатываемой библиотеки работы с вводом-выводом

3.1 Схема применения библиотеки

Работа с библиотекой начинается с вызова функции инициализации библиотеки. Реализация функции инициализации может быть архитектурно зависима. Например, для системы BlueGene /P в момент инициализации библиотеки используются функции из библиотеки MPIX[5] для определения коммуникаторов, соответствующих множествам вычислительных узлов использующих один или наоборот разные узлы ввода-вывода. Исходя из полученных коммуникаторов, вычисляются возможные вычислительные узлы ввода-вывода. Функция инициализации должна вызываться на всех узлах, выделенных программе, т.е. в рамках коммуникатора MPI_COMM_WORLD.

Дальнейшую работу библиотеки пользователь может продолжить на всех процессорах, или указать число процессоров или коммуникатор, в рамках которого будет происходить работа библиотеки. Пользователю предоставляется возможность создать несколько независимых областей работы библиотеки в рамках подкоммуникаторов. В частности, библиотека может использоваться в системах визуализации данных [7].

В рамках каждого коммуникатора библиотеки возможно указывать источники данных, для этого предусмотрены функции вида «ParimprC_DataLoad_[Type]()», где в качестве типа источника указывается например netCDF или HDF; системный программист может добавлять источники данных добавляя функции подобного вида. Данные из источника загружаются на вычислительные узлы ввода-вывода, после чего доступны через поля структуры в списке всех источников данных.

С помощью функции «ParimprC_LayoutCreate()» может быть выделена область обработки. В зависимости от того, сколько MPI-процессов будет принадлежать области обработки, пользователь может создать множество независимых областей обработки. Системный программист может дорабатывать алгоритм данной функции, изменяя стратегию объединения MPI-нитей в области обработки. В зависимости от стратегии объединения, пользователь получит область обработки из соседних или удаленных вычислительных узлов. Указывая область обработки и типа распределения данных между узлами, пользователь с помощью функции «ParimprC_DataPrepare()» распределяет данные между вычислительными узлами области обработки.

В рамках области обработки могут быть вызваны функции-обработчики данных. Пользователь может создавать свои обработчики данных или использовать доступные в библиотеке. Обработчик данных работает в рамках MPI-коммуникатора, отвечающего области обработки.

3.2 Представление данных

В библиотеке определено абстрактное понятие «источник данных», для которого определен способ загрузки данных на узлы вычислительной машины. Данные могут иметь разный

формат и способ представления. В рамках данного этапа возможности библиотеки распространяются на работу с данными, представимыми в виде многомерных массивов.

Наиболее популярными в научной среде форматы хранения многомерных массивов во внешней памяти являются форматы NetCDF и HDF. Для этих форматов существует интерфейс взаимодействия на различных языках программирования, в частности C++.

Загрузка данных из файлов таких форматов происходит в три этапа:

- открытие файла
- определение имени и/или пути к данным внутри файла
- указание подмножества данных для загрузки

В результате для файла с массивом размерности N можно извлекать любые подмассивы размерности ≤ N. Требуется универсальное представление внутри оперативной памяти для массивов любой размерности.

Существует несколько решений для представления многомерного массива основанных на шаблонах, в параметрах шаблона указывается тип, размерность и максимальные значения индексов. Недостатком данного решения является то, что тип элементов массива определяется статически и невозможно изменить его в зависимости от того, какие данные были обнаружены в источнике данных в процессе выполнения программы.

Более универсальное решение — представление данных многомерного массива в виде проекции на одномерный массив типа char (или минимальной адресуемой единицы).

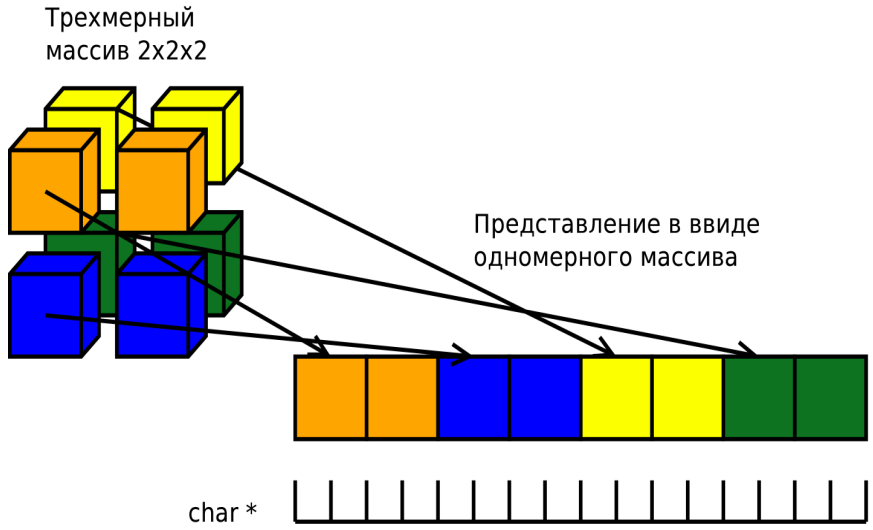

Рис. 3. Представление трехмерного массива

Разбиение данных на части по последнему измерению многомерного массива сводиться к разделению одномерного массива на последовательные части. Обработка произвольного подмассива произвольной размерности (меньшей изначальной) задача более сложная, требуются дополнительные операции, если функции обработки предполагают наличие последовательно расположенных элементов.

Для передачи данных средствами MPI определены стандартные типы данных, для передачи массива указывается тип данных и количество передаваемых элементов. Когда мы хотим передать произвольный подмассив многомерного массива, такой способ не подходит, так как данные не обязательно расположены последовательно.

Для подобного представления массива в MPI определены дополнительные типы. Для получения такого типа используется следующая MPI-функция: MPI_Type_create_subarray()

3.3 Выбор узлов ввода-вывода

Первая подзадача, которую требуется решить — выбор узлов ввода-вывода.

Каждый узел в вычислительной системе имеет ограниченное число оперативной памяти. Размер данных в источнике данных может многократно превосходить объем оперативной памяти отдельного вычислительного узла. Данные из источника распределяются по оперативной

памяти узлов ввода-вывода, от количества узлов ввода-вывода зависит возможный размер данных. Размер загружаемых данных определяется задачей и ее параметрами, поэтому заранее установить конкретное требуемое число узлов ввода-вывода на основе информации о вычислительной системе. Учитывая все вышесказанное, было решено оставить решение вопроса о количестве узлов ввода-вывода за пользователем. Во время инициализации библиотеки пользователь может указать, какое именно число процессоров он хочет задействовать в качестве узлов ввода-вывода.

После указания количества узлов ввода-вывода необходимо определить, какие именно вычислительные узлы будут отвечать за операции ввода-вывода. Подобный выбор может существенно повлиять на производительность дальнейшей загрузки и выгрузки данных.

Для системы BlueGene /P [3], установленной на факультете вычислительной математики и кибернетики МГУ имени М.В. Ломоносова, 128 вычислительных узлов используют один канал ввода-вывода. Если пользователю доступно для вычислений более 128 вычислительных узлов, то для выбора узлов ввода-вывода требуется определить группы процессоров, которые отвечают одному каналу ввода-вывода.

В MPI-реализации на BlueGene /P доступны три дополнительные функции, расширяющие функциональность механизма передачи сообщений с учетом особенностей аппаратуры системы. Они используются в рамках коммуникаций по сети трехмерного тора и упрощают отображение процессов на наборы процессоров.

Коллективная операция MPIX_Pset_same_comm_create (MPI_Comm *pset_comm) создает набор коммуникаторов, в каждый из которых входят все вычислительные узлы из данного набора процессоров. В результате обращения к коллективной функции MPIX_Pset_diff_comm_create(MPI_Comm *pset_comm) будут созданы коммуникаторы, объединяющие узлы, принадлежащие разным наборам процессоров, т.е. никакие два процесса каждого коммуникатора не будут иметь общего для них узла ввода-вывода.

Для выбора узлов ввода-вывода в разрабатываемой библиотеке определен алгоритм, который последовательно выбирает для узлов ввода-вывода процессы, для которых порядковый номер в коммуникаторах MPIX_Pset_same_comm_create() - минимален. При этом количества узлов ввода-вывода соответствующих одному каналу ввода-вывода различаются не более чем на единицу.

Выбранные узлы ввода-вывода объединяются в новый MPI-коммуникатор. Вызов функций загрузки-выгрузки данных может быть вызван на любом вычислительном узле, но на узлах не относящихся к узлам ввода-вывода никакой работы выполнено не будет. Такая модель защищает от ошибок вызова операций загрузки данных в рамках неправильного MPI-коммуникатора.

3.4 Выбор вычислительных узлов обработки данных

Пользователю библиотеки предлагается выбрать один из нескольких режимов работы:

1. Работа библиотеки на всех доступных задаче узлах вычислительной системы
2. Работа библиотеки в рамках отдельного коммуникатора

Первый и второй случай выглядят одинаково с точки зрения программы, поскольку запуск на всех доступных задачи узлах вычислительной системы равносилен запуску библиотеки в рамках глобального MPI-коммуникатора *MPI_COMM_WORLD*.

Другой принцип разделения режимов основан на выборе узлов обработки. Для запуска функций обработки пользователь должен определить коммуникатор, в рамках которого будет происходить распределение данных и непосредственно обработка распределенных данных. Процесс определения коммуникатора с вычислительными узлами обработки будем называть выделением области обработки. Существует два принципиально различных способа выбора областей обработки:

1. Области обработки выделяются с использованием вычислительных узлов, не задействованных в операциях ввода-вывода.
2. Области обработки выделяются, в том числе, с использованием вычислительных узлов отвечающих за ввод-вывод.

На данном этапе реализация библиотеки предполагает, что первый подход (использование узлов ввода-вывода для вычислений) используется только в том случае, когда обработка происходит на всех доступных задаче вычислительных узлах, иначе говоря, в рамках *MPI_COMM_WORLD*.

Пользователь может указать произвольный MPI-коммуникатор в качестве области обработки (MPI-коммуникатор не должен содержать узлов ввода-вывода).

3.4 Распределение данных на вычислительные узлы

После определения вычислительных узлы, на которых будет происходить обработка, необходимо расположить данные так, как предполагает их получать функция обработки данных. Возможны две стратегии решения указанной подзадачи:

1. Функции обработки разрабатываются так, что обладают встроенным параметром, в котором указано, какие данные должны находиться на каком вычислительном узле обработки.
2. Существует дополнительная функция, которая позволяет пользователю указывать желаемое распределение данных по узлам вычислительной системы.

Для данных в виде массива размерности N пользователь может указать список, в котором для каждого вычислительного узла будут указаны диапазоны координат или координаты отдельных элементов, все координаты указываются для N-мерного случая. Список должен иметь длину не больше, чем число MPI-нитей в области обработки, для которой происходит перераспределение данных. В соответствии с этим списком данные будут распределены по узлам вычислительной системы, предназначенным для обработки.

Поскольку, в общем случае, данные из источника распределены на узлах ввода-вывода независимо от последующего их использования в программе, процесс передачи определенного подмножества данных из источника на определенный узел обработки данных может потребовать передачи данных от нескольких узлов ввода-вывода.

Пример:

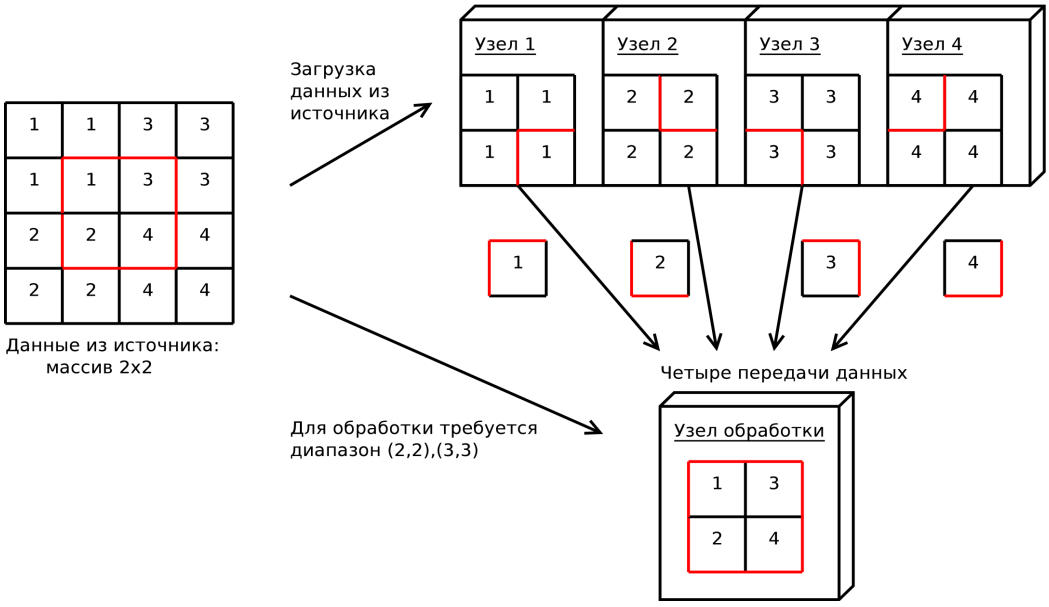

Рис. 4. Кусочное распределение данных

Данные из источника (двухмерный массив целых чисел) делится на равные порции и загружается на четыре узла ввода-вывода. Узлу обработки требуется подмассив из элементов (2,2), (2,3), (3,2), (3,3): потребуется четыре передачи данных. Стандартные MPI-функции передачи данных предполагают вызов функции приема данных для каждой порции. При этом требуется указывать буфер приема определенного размера, хотя порции данных от разных узлов ввода-вывода могут иметь разный размер.

Существует две возможности реализации подобного алгоритма:

1. Узлы обработки знают конкретное расположение данных на узлах ввода-вывода, тогда при узел обработки будет последовательно принимать данные от узлов ввода-вывода. Поскольку изначально предполагается, что узлы ввода-вывода могут работать независимо от узлов обработки, продолжая загрузку данных одновременно с обработкой, такой вариант потребует накладных расходов на извещение узлов обработки о конкретном расположении данных на узлах ввода-вывода.
2. Перед каждой передачей данных узел ввода-вывода извещает узел обработки о характеристиках последующей передачи (какие именно данных содержит узел ввода-вывода, размер данных). Такой способ создает дополнительную нагрузку на систему коммуникаций.

Для распределения данных на вычислительные узлы обработки используется функция односторонней передачи данных MPI_Put( ). Операция по сбору данных с узлов обработки на узлы ввода-вывода осуществляется с помощью MPI-функции MPI_Get().

После распределения на вычислительных узлах обработки доступны списки тех диапазонов данных, которые были указаны в функции распределения.

4. Анализ эффективности

Как описывалось выше, существует множество стратегий использования разрабатываемой библиотеки для обработки данных. Выше приведены аналитические оценки характеристик идеи выделения вычислительных узлов, отвечающих за все операции ввода-вывода. Необходимо исследовать практичность подхода на экспериментальных задачах.

Прежде всего необходимо определить характеристики по которым будет происходить сравнение. Конечным параметром сравнения разных подходов одинаковой обработки одни и тех же данных будет являться время выполнения.

В качестве функции обработки данных будет использована синтетическая тест-функция, которая принимает на вход несколько численных параметров: количество сложений, количество вычитаний, количество умножений и количество делений. Данная функция будет производить указанное число операций над каждым элементом многомерного массива, который был назначен на вычислительный узел обработки данных.

Поскольку требуется сравнить подход с выделением узлов ввода-вывода и обычный подход, когда все узлы могут использовать операции ввода-вывода, были разработаны дополнительные функции для формата HDF5, позволяющие вычислительному узлу получить произвольные данные из файла в указанном формате.

Экспериментальные данные создаются при помощи разработанного средства генерации многомерных массивов требуемого размера со случайными числовыми значениями его элементов. Данное средство генерации сохраняет выходные данные в виде одного файла в формате HDF5.

На основе разрабатываемой библиотеки, дополнительных функций и генератора экспериментальных данных было создано несколько режимов тестирования:

1. Полное использование библиотеки Parimpr с обработкой на всех узлах:
- Открытие файла, загрузка данных на узлы ввода-вывода, перераспределение данных на все вычислительные узлы, доступные тестовой задаче, происходит с использованием функций библиотеки.
- Запуск тестовой функции обработки данных происходит на всех вычислительных узлах, доступных тестовой задаче.
- После обработки результат собирается на узлах ввода-вывода и записываются в файл
2. Использование библиотеки Parimpr только для загрузки данных с обработкой на всех узлах:
    - Открытие файла, загрузка данных на узлы ввода-вывода, перераспределение данных на все вычислительные узлы, доступные тестовой задаче, происходит с использованием функций библиотеки.

- Запуск тестовой функции обработки данных происходит на всех вычислительных узлах, доступных тестовой задаче.
- Запись в файл происходит прямо с узлов обработки данных

3. Без использования библиотеки Parimpr:

- Файл открывается, данные загружаются непосредственно на узлах обработки данных, дополнительное распределение данных не требуется.
- Запуск тестовой функции обработки данных происходит на всех вычислительных узлах, доступных тестовой задаче.
- Запись в файл происходит прямо с узлов обработки данных

4. Полное использование библиотеки Parimpr без обработки данных на узлах ввода-вывода:

- Открытие файла, загрузка данных на узлы ввода-вывода, перераспределение данных на все вычислительные узлы не участвующие в операциях ввода-вывода, доступные тестовой задаче, происходит с использованием функций библиотеки.
- Обработка запускается только в выделенной области обработки
- После обработки результат собирается на узлах ввода-вывода и записывается в файл

5. Дополнение режимов 3 и 4: обработка потока данных. Операции, указанные в режимах 3 и 4, выполняются многократно над последовательностью файлов с данными, при этом для режима 4 предусмотрена загрузка данных из следующего файла одновременно с выполнением обработки текущих данных на остальных вычислительных узлах обработки данных.

Для каждого из режимов тестирования считается отдельное время работу загрузки данных, обработки и выгрузки.

Для каждого из режимов тестирования необходимо рассмотреть график зависимости общего времени работы тестовой задачи в зависимости от следующих параметров:

- размер исходных данных;
- количество операций в тестовой функции обработки данных.

Заключение

В данной работе рассмотрены подходы к организации ввода-вывода в задачах с потоком входных-выходных данных и высокой загрузкой каналов, соединяющих вычислительные узлы и внешние системы хранения данных. Разработаны методы разделения функциональности вычислительных узлов в рамках модели иерархии коммуникаторов. Авторами разработана библиотека, реализующая описанные методы. Исследование производительности разработанной библиотеки проводилось на системах Blue Gene/P и «Ломоносов».

Литература

1. NetCDF Tutorial
   URL: http://www.unidata.ucar.edu/software/netcdf/docs/ (дата обращения: 10.02.2011)
2. Сайт BlueGene /P, установленного в МГУ
   URL: http://hpc.cs.msu.su (дата обращения: 10.02.2011)
3. James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C. Charles Law, Michael Papka. Large-Scale Data Visualization Using Parallel Data Streaming. // IEEE Computer Graphics and Applications, July — Augus, 2001/ Vol. 21. No. 4. P. 34-41.
4. Alok Choudhary, Wei-keng Liao, Kui Gao, Arifa Nisar, Robert Ross, Rajeev Thakur and Robert Latham. Scalable I/O and analytics // Journal of Physics: Conference Series, 2009. Vol 180. No 1.

5. Hongfeng Yu, Kwan-Liu Ma, Joel Welling: I/O Strategies for Parallel Rendering of Large Time-Varying Volume Data. // 5th Eurographics/ACM SIGGRAPH Symposium on Parallel Graphics and Visualization (EGPGV 2004) ,June, 2004. P. 31-40, .
6. Библиотека mpix.
URL: http://www.mpix.com (дата обращения: 10.02.2011)
7. Джосан О.В., Мурынин А.Б., Попова Н.Н. Метод визуализации многомерных динамических данных на многопроцессорных комплексах // Вестник компьютерных и информационных технологий. 2009. № 8. сс. 8-12.
8. Корж А.А., Макагон Д.В., Оценка минимальных требований к аппаратуре и топологии при построении высокоскоростных коммуникационных сетей для суперкомпьютеров с общей памятью // Вычислительные методы и программирование: новые вычислительные технологии. 2008. Т. 9. № 2. сс. 26-31.