DOI: 10.15514/ISPRAS-2023-35(2)-8

Пути организации параллельного доступа к структурированным данным

А.О. Игнатьев, ORCID: 0000-0003-4902-2123 <a.o.ignatyev@vniitf.ru>
С.Ю. Мокшин, ORCID: 0000-0002-7454-6597 <s.yu.mokshin@vniitf.ru>
Д.В. Иванков, ORCID: 0000-0003-4254-0104 <d.v.ivankov@vniitf.ru>
Е.А. Бекетов, ORCID: 0009-0008-0766-6258 <e.a.beketov@vniitf.ru>

Всероссийский НИИ технической физики имени академика Е.И. Забабахина,
456770, Россия, г. Снежинск, Челябинская область, ул. Васильева, 13

Аннотация. В данной работе исследуются пути достижения максимально возможной производительности обменов с файлами, содержащими структурированные данные. Исследования проводились на файловых системах с параллельным доступом вычислительных систем, предназначенных для решения задач физико-математического моделирования различных процессов и объектов. На примере параллельной файловой системы Lustre рассматривается параллельный доступ к «сырым» данным. Предлагается способ организации параллельного доступа к структурированным данным на основе специального разработанного формата хранения PSIO и библиотеки доступа psio. Выполняется сравнительный анализ производительности ввода-вывода разработанного формата хранения данных и формата параллельной версии HDF5.

Ключевые слова: вычислительная система; структурированные данные; высокопроизводительные вычисления; математическое моделирование; форматы данных; параллельный ввод/вывод.

Для цитирования: Игнатьев А.О., Мокшин С.Ю., Иванков Д.В., Бекетов Е.А. Пути организации параллельного доступа к структурированным данным. Труды ИСП РАН, том 35, вып. 2, 2023 г., стр. 111-126. DOI: 10.15514/ISPRAS–2023–35(2)–8

Благодарности: Авторы публикации выражают особую благодарность Р.М. Шагалиеву и В.А. Свиридову за предоставленные для проведения тестов вычислительные ресурсы Полигона НЦФМ (г. Саров).

Ways to organize parallel access to structured data

A.O. Ignatyev, ORCID: 0000-0003-4902-2123 <a.o.ignatyev@vniitf.ru>
S.Yu. Mokshin, ORCID: 0000-0002-7454-6597 <s.yu.mokshin@vniitf.ru>
D.V. Ivankov, ORCID: 0000-0003-4254-0104 <d.v.ivankov@vniitf.ru>
E.A. Beketov, ORCID: 0009-0008-0766-6258 <e.a.beketov@vniitf.ru>

E. I. Zababakhin All-Russian Scientific Research Institute of Technical Physics,
13, Vasilieva street, Chelyabinsk region, Snezhinsk, 456770, Russia

Abstract. This paper explores ways to achieve the highest possible exchange performance with files containing structured data. The research was carried out on file systems with supercomputer systems parallel access designed to solve problems of physical and mathematical modeling of various processes and objects. For example, parallel access to raw data is considered using the Lustre file system. The article suggests a way to organize parallel access to structured data based on a specially developed PSIO storage format and the psio access library. A comparative analysis of the I/O performance of the developed data storage format and the HDF5 parallel version format is performed.

Keywords: computing system; structured data; high-performance computing simulation; mathematical simulation subsystem; data format; HDF; HPC; SIO.

For citation: Ignatyev A.O., Mokshin S.Yu., Ivankov D.V., Beketov E.A. Ways to organize parallel access to structured data. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 2, 2023. pp. 111-126 (in Russian). DOI: 10.15514/ISPRAS-2023-35(2)-8

Acknowledgements. The authors express special gratitude to R.M. Shagaliev and V.A. Sviridov for the NCFM Polygon computing resources provided for the tests (Sarov city).

1.Введение

Развитие и усложнение цифровых моделей различных физических объектов неизбежно приводит к увеличению объемов обрабатываемой на вычислительных системах (ВС) информации, а также широкому использованию параллельных файловых систем (ПФС) и механизмов массового распараллеливания прикладных программ, использующих структурированные данные, что, в свою очередь, неизбежно ставит вопрос о повышении скорости доступа к этим данным.

Простой путь увеличения производительности доступа к данным заключается в том, что данные каждого процесса расчетной задачи размещаются в отдельном структурированном файле, таким образом, доступ к ним осуществляется в два этапа:

- сначала определяется имя файла исходя из номера процесса;
- потом, по соответствующему запросу, выбирается блок данных из этого файла (например, с использованием функционала HDF5 [1]).

Для хранения объектов также можно использовать отдельные файлы, в имя которых входит номер процесса и имя объекта.

Таким образом, при таком подходе, структурированный набор данных представляется множеством файлов в общем каталоге. Благодаря встроенному в ПФС механизму размещения данных среди группы устройств хранения ПФС, все входящие в этот структурированный набор файлы более-менее равномерно распределяются по всем Objects Store Target (OST) ПФС Lustre [2]. Такая технология называется «файл на процесс» и получила распространение во всем мире, а также широко используется в РФЯЦ-ВНИИТФ. Эта же технология используется, как одна из возможных, при определении производительности ПФС на тестах IO-500 [3].

Производительность при этом подходе обеспечивается базовыми возможностями параллельной файловой системы (ПФС), благодаря автоматическому распределению потоков ввода/вывода между всеми доступными устройствами хранения ПФС.

При очевидной простоте этого подхода он имеет несколько недостатков:

- двухэтапная схема адресации объектов: сначала имя файла, потом объект в файле;
- существенное увеличение количества файлов в ПФС, прямо пропорциональное количеству процессов в задаче, создающее повышенную нагрузку на метасервис ПФС, что отрицательно сказывается на качестве функционирования файловой системы.

Другой путь реализации заключается в заложенных в ПФС возможностях по параллельному доступу процессов задачи к одному файлу. Так устроен механизм MPI-IO [4], используемый в основе современных библиотек структурированного ввода/вывода типа HDF5.

Обширность темы эффективного параллельного ввода-вывода не позволяет рамками отдельной статьи охватить всё многообразие используемых методов и инструментов. Поэтому остановимся на рассмотрении основных принципов и выявлении характерных проблем параллельного ввода-вывода, с которыми могут сталкиваться начинающие разработчики High Performance Computing (HPC) приложений, а также приведём примеры программ, реализующих различные методы параллельного доступа к данным в файловой системе Lustre.

Для этого вначале рассматриваются общие принципы и возможности параллельного доступа к данным на ПФС на примере Lustre, делаются предположения об организации параллельного доступа к структурированным файлам, потом производится проверка на модельной версии библиотеки psio и HDF5.

2. Проверка базовых возможностей Lustre

Параллельная файловая система Lustre предполагает для организации параллельного доступа к наборам данных использовать стрипование (разделение) файлов, при котором файл разбивается на сегменты одинакового размера, определяемые атрибутом stripe_size (по умолчанию 1048576 байт или 1 МБ). Сегменты файла распределяются между OST (Objects Store Target – устройство хранения) ПФС Lustre. Число задействованных OST определяется атрибутом stripe_count, задается при создании файла и не должно превышать число используемых в ПФС OST. Тем самым, файл с stripe_count = 1 является обычным файлом с последовательным доступом. Файл же с stipe_count больше 1 допускает параллельный доступ к сегментам, расположенным на разных OST. При задании stripe_count, равным 0, файл распределяется на все доступные OST.

На рис. 1 показана общая схема стрипования файла для ПФС, содержащей n OST. Файл содержит m групп сегментов по n сегментов в группе. Цвет сегмента определяет размещение его на соответствующем OST.

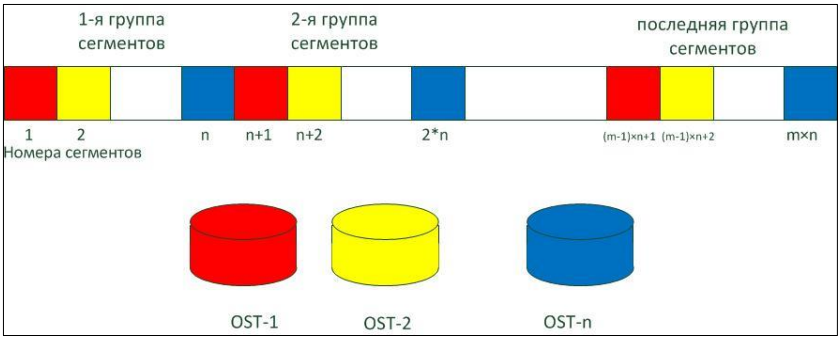

Рис. 1. Общая схема стрипования файлов
Fig.1. The general scheme of file stripping

2.1 Исходные данные

Для проверки возможностей параллельного доступа к стрипованному файлу на вычислительном комплексе (ВК) «Полигон НЦФМ» (г. Саров) [5] создано тестовое пространство на ПФС Lustre, производительность которой на тесте IOR из состава набора тестов IO-500 [3] на операции записи составила 9.447 ГБ/с при общем объеме данных 2 ТБ и отсутствии посторонней нагрузки. Этот показатель будем считать максимальной производительностью ПФС и использовать в дальнейшем для сравнения.

2.2 Методика оценки производительности параллельного ввода-вывода

2.2.1 Оценка пропускной способности одного канала доступа к ПФС

Важным показателем при исследовании параллельности обменов с ПФС является ускорение, полученное при использовании множества потоков по отношению к одному, поэтому для нас

интересна как пропускная способность одного канала (используется один поток данных), так и пропускная способность ПФС при использовании нескольких потоков.

Для оценки пропускной способности одного канала осуществляется последовательная запись в файл с атрибутом stripe_count=1 с одного процесса задачи блоками большой длины, с выталкиванием файловых кэшей на стороне клиента после записи каждого блока, как показано на рис. 2. Скорость записи определяется как отношение суммарного объема записываемых блоков к времени записи. Второй процесс задачи на другом вычислительном узле (чтобы исключить влияние файловых кэшей на стороне клиента) последовательно читает эти блоки из того же файла. Скорость чтения определяется как отношение суммарного объема прочитанных блоков к времени чтения.

Рис. 2. Схема определения пропускной способности канала ПФС
Fig.2. The scheme of determining Lustre channel bandwidth

2.2.2 Оценка пропускной способности ПФС Lustre на параллельных обменах

Для оценки пропускной способности ПФС при использовании нескольких потоков на первом шаге со стороны всех процессов параллельной задачи производится запись данных в файл с stripe_count, отличным от 1, как показано на рис. 3. Количество используемых потоков (процессов) выбирается равным или кратным атрибуту stripe_count. Запись организуется в цикле, до достижения требуемого объема передаваемого с процесса информации. При этом каждому процессу назначаются соответствующие области в файле. На первой итерации этого цикла каждый процесс пишет в файл блок размера stripe_size со смещением, равным порядковому номеру процесса (его ранку в параллельной задаче), умноженному на stripe_size. На следующих итерациях смещение увеличивается на объем записанной информации (stripe_size × nproc, где nproc – число процессов в параллельной задаче). На втором шаге производится аналогичным способом чтение блоков данных из этого файла, при этом, для исключения влияния файловых кэшей на стороне клиента, отображение процессов задачи на блоки данных в файле производится со смещением:

- процесс 0 читает блок 1, записанный на процессе с ранком 1,
- процесс 1 читает блок 2, записанный на процессе с ранком 2,
- последний процесс читает блок 0, записанный на процессе с ранком 0.

Рис. 3. Схема определения пропускной способности ПФС
Fig.3. The scheme of determining Lustre multi-channel bandwidth

2.2.3 Результаты замеров на неструктурированных данных

Для тестирования производительности ввода-вывода использовалась тестовая программа, позволяющая создавать файл при запуске программы с разными атрибутами. При тестировании выполнялось следующее:

- при каждом запуске файл создавался заново с атрибутом stripe_count, равным числу потоков данных (пока это число меньше количества OST), или равным количеству OST (в случае если это число больше количества OST);
- для каждого числа потоков на обмен подавались блоки размером 1, 2, 4, 8, 16 и 32 МБ (в таблице колонка BS) с сохранением суммарного объема информации в потоке равным 96 МБ, при этом использовалось 96, 48, 24, 12, 6 и 3 итераций (в таблице колонка Nit);
- параметр stripe_size (в таблице колонка SS) при создании файла принимался равным BS;
- каждый тест запускался не менее трех раз, при этом в таблицы приведены лучшие показатели: Vw – производительности на записи, Kw – ускорение записи по отношению с усредненной производительности на одном потоке, Vr – производительности на чтении, Kr – ускорение чтения по отношению с усредненной производительности на одном потоке.

В табл. 1 приведены результаты, полученные по этой методике на ВК. Выделены лучшие результаты для каждой конфигурации прогона.

Пропускная способность канала на ВК составила 655.6 МБ/с и достигается на блоках в 32 МБ с соответствующим размером сегмента. Увеличение размера блока положительно влияет на производительность, поэтому для числа потоков больше четырех использовались только блоки размером 8, 16 и 32 МБ.

Табл. 1. Пропускная способность ПФС на ВК
Table 1. Cluster Lustre bandwith

BS	Nit	SS	Vw	Vr	Kw	Kr
1 поток, размер 96 МБ						
1	96	1	293.8	865.2		
2	48	2	342.8	772.8		
4	24	4	395.9	740.9		

8	12	8	504.9	805.0		
16	6	16	596.8	752.9		
32	3	32	655.6	786.3		
2 потока, размер 192 МБ						
1	96	1	575.1	1387.1	1.3	1.7
2	48	2	678.1	1301.3	1.5	1.6
4	24	4	804.3	1428.1	1.8	1.8
8	12	8	974.5	1407.0	2.1	1.8
16	6	16	1183.4	1412.1	2.6	1.8
32	3	32	1300.2	1612.8	2.8	2.0
4 потока, размер 384 МБ						
8	12	8	2146.4	2839.4	4.7	3.6
16	6	16	2364.2	2857.7	5.2	3.6
32	3	32	2580.4	3005.0	5.6	3.8
8 потоков, размер 768 МБ						
8	12	8	4446.9	6483.6	9.7	8.1
16	6	16	5077.6	6300.7	11.1	7.9
32	3	32	5731.9	6615.5	12.5	8.3
16 потоков, размер 1536 МБ						
8	12	8	8063.7	11635.0	17.6	14.6
16	6	16	9597.2	11487.8	20.9	14.4
32	3	32	8991.5	11864.4	19.6	14.9
32 потока размер 3072 МБ						
8	12	8	9001.8	20444.3	19.6	25.7
16	6	16	9495.7	21090.4	20.7	26.5
32	3	32	8159.2	21946.6	17.8	27.5

Максимальная скорость записи на ВК составила 9.49 ГБ/с на 32 потоках с блоком 32 МБ, что соответствует производительности на тесте IOR (9.45 ГБ/с). Ускорение по отношению к усредненному однопоточному результату составило 20.7 раза.

3. Моделирование параллельного ввода-вывода структурированных данных

3.1 Особенности работы со структурированными данными

Структурированные файлы, в общем случае, содержат каталог и блоки данных. Каталог содержит описание блоков данных (идентификатор блока данных, тип данных в блоке, размер блока и смещение по файлу для размещения блока).

Поскольку максимальная производительность обмена процесса с ПФС достигается в том случае, когда потоки данных с процессов на OST не пересекаются по OST, необходимо обеспечить выравнивание блоков данных по границам сегментов файла, т.е. каталогу отвести один или более целых сегментов и каждый блок данных размещать с границы очередного сегмента. Кажущееся увеличение размера файла, за счет пустого пространства в конце каждого сегмента по сравнению с плотной упаковкой данных в файле или представлением набора данных в виде множества файлов, компенсируется тем, что ПФС выделяет дисковое пространство под файлы с точностью до размера сегмента.

Рассмотрим простейшую схему организации структурированного набора данных и работы с ним:

- набор данных содержит в себе именованные блоки данных;
- каждый блок данных представляет собой одномерный массив чисел;
- все блоки данных описываются каталогом, являющимся неотъемлемой частью набора данных;

- для каждого блока данных в каталоге присутствуют имя блока, число и формат элементов, смещение по файлу или индекс сегмента в файле (или множество индексов для блоков данных, размер которых превышает размер сегмента);
- выделяемые сегменты описываются в таблице (карте) сегментов, которая, наравне с каталогом, является метаинформацией структурированного набора данных;
- порядок описания блоков данных в каталоге не существенен;
- для операции чтения/записи блока данных требуется заранее открытый файл, имя блока данных и массив для обмена информацией в оперативной памяти процесса задачи;
- все операции с набором данных являются коллективными в смысле MPI, т.е. одновременно выполняются на всех процессах, входящих в коммуникационную группу;
- во избежание конфликтов по доступу к OST со стороны процессов задачи, все сегменты одного блока данных должны располагаться на одном OST;
- операции, модифицирующие каталог, выполняются на ранке 0 задачи по групповым запросам от остальных ранков, модифицированный каталог рассылается всем ранкам задачи.

Предлагаемая схема работы с данными набора со стороны задачи идентична схеме, описанной в п. 2.2.1 и 2.2.2 (рис. 2 и 3), за исключением того, что доступ к данным в файле производится не по индексу в файле, а по имени блока данных.

3.2 Реализация работы со структурированными файлами

Для проверки работы со структурированными файлами была написана библиотека psio, содержащая следующие функции:

- PSIO_Fop – открытие (создание) стрипованного файла на ФС Lustre;
- PSIO_Fcl – закрытие файла;
- PSIO_Bwr – запись именованного блока данных;
- PSIO_Brd – чтение именованного блока данных;
- PSIO_Fcat – вывод каталога файла (имен и характеристик блоков данных).

Файл формата PSIO имеет следующую структуру:

- вся информация в файле содержится в сегментах, размером stripe_size байт, и индексируется номером сегмента,
- каждый объект в файле представлен набором сегментов;
- файл содержит каталог и карту сегментов, содержащую пары: индекс объекта и номер сегмента объекта;
- каталогу присваивается номер объекта 0;
- карте сегментов присваивается номер объекта 1;
- блокам данных присваиваются следующие номера объектов;
- каждый блок данных в файле записывается сегментами и представлен записью в каталоге, в которой указывается имя блока, формат и количество элементов в нем, количество сегментов в файле;
- всем сегментам блока выделяется сегменты файла, расположенные на одном OST.

На листинге 1 приведен пример программы, записывающий с каждого ранка параллельной задачи по одному блоку данных двойной точности длиной 8 МБ в файл, стрипованный на все OST с размером сегмента 2 МБ.

```
1 #include <mpi.h>
2 #include <psio.h>
3 PSIO_HF *hf;                    /* дескриптор файла */
```

```
 4 int stripe_count = 0;         /* на все OST */
 5 int stripe_size = 2*1024*2014;      /* сегмент 2 МБ */
 6 int main(int argc, char** argv)
 7 {
 8   int iRank, iSize;
 9   char bname[20];
10   int blen = 1024*2014;
11   double array;
12   /* Инициализация MPI */
13   ierr  = MPI_Init(&argc, &argv);
14   MPI_Comm_rank( MPI_COMM_WORLD, &iRank);
15   MPI_Comm_size( MPI_COMM_WORLD, &iSize);
16   /* Формирование опций для создания файла */
17   sprintf(opts, "stripe_count=%d, stripe_size=%d stripe_offset=-1",
            stripe_count, stripe_size);
18   /* Создание файла для параллельного доступа */
19   if (PSIO_Fop(fname, "new", &hf, opts, MPI_COMM_WORLD))  exit(-1);
20   /* Запись блоков данных */
21   sprintf(bname, "B%03d", iRank);
22   array = GetData(iRabk, blen);      /* Генерация 8 МБ данных  */
23   if (PSIO_Bwr(hf, bname, "real*8", blen, array, MPI_COMM_WORLD))
       exit(-1);
24   /* Вывод каталога файла */
25   PSIO_Fcat(hf);
26   /* Закрытие файла */
27   if (PSIO_Fcl(hf, MPI_COMM_WORLD)) exit(-1);
28   /* Завершение MPI */
29   MPI_Finalize();
30   return 0;
31 }
```

Листинг 1. Пример исходного текста программы, выполняющей запись данных в параллельном режиме

Listing 1. An example of the source code of a program that writes data in parallel mode

Эта тестовая программа выполняет обмены со структурированным файлом, при этом выполняются следующие действия:

- файл открывается на запись (строка 19),
- все процессы в цикле по указанному в параметрах числу итераций производят запись в файл блоков данных заданной длины. Имя блока строится на основании ранка процесса и номера итерации (строки 22-23);
- файл закрывается (строка 25).

Каждая функция библиотеки оформлена как коллективная операция в смысле MPI, при этом:

- чтение каталога файла и карта сегментов при его открытии выполняется на ранке 0 задачи, после этого каталог и карта сегментов рассылается остальным процессам;
- при записи нового блока с процессов задачи запрос на включение блока в каталог файла отсылается на ранк 0 задачи, где выполняется операция по модификации каталога и карты сегментов, после этого модифицированная метаинформация рассылаются остальным процессам задачи;
- при закрытии модифицированного файла сохранение каталога и карты сегментов производится с ранка 0;
- все взаимодействие между процессами задачи для обеспечения одинаковости метаинформации производится средствами MPI.

3.3 Полученные результаты

Поскольку каждая операция с файлом, кроме обменов с ПФС, содержит обращения к функциям MPI, создающие дополнительные накладные расходы, при тестировании использовались блоки разной длины, размещаемые в нескольких сегментах файла. Скорость выполнения обмена определялась как отношение полного объема данных к времени выполнения цикла записи/чтения с учетом времени, необходимого на открытие и закрытие файла.

Поскольку в библиотеке psio блоки данных пишутся сегментами с размером, равным атрибуту stripe_size, при тестировании варьировались как размер блока, так и размер сегмента, при этом размер передаваемой на потоке информации увеличен до 1 ГБ.

Проведена следующая серия замеров:

- сначала на одном и двух потоках проверялось влияние размера блока и размера сегмента при больших блоках на скорость записи, при этом обнаружено, что при размере блока в 512 МБ и размере сегмента 8 МБ наступает насыщение;
- затем, для блоков в 128, 256 и 512 МБ и сегменте 8 МБ проведены расчеты на 10, 48 и 96 потоках данных.

В табл. 2 приведены результаты, полученные на ВК.

Получена максимальная скорость записи 9.885 ГБ/с на 32 потоках данных блоками по 512 МБ и размере сегмента 8 МБ, что даже лучше, чем результаты, полученные на тесте IOR и неструктурированных данных. Увеличение производительности, видимо, связано с увеличением объема данных, передаваемых за одну операцию.

Табл. 2. Пропускная способность ПФС на структурированных данных на ВК
Table 2. Cluster Lustre bandwith on a structured data

BS	Nit	SS	Vw	Vr	Kw	Kr
1 поток, размер 1024 МБ						
1	1024	1	319.3	767.7	0.8	0.9
2	512	1	325.9	726.0	0.8	0.9
4	256	1	314.3	735.7	0.8	0.9
8	128	1	320.7	742.7	0.8	0.9
16	64	1	327.3	721.6	0.8	0.9
32	32	1	319.6	717.2	0.8	0.9
64	16	1	324.2	777.1	0.8	1.0
128	8	1	324.7	727.1	0.8	0.9
128	8	2	395.9	776.0	1.0	1.0
128	8	4	457.9	828.6	1.2	1.0
128	8	8	602.9	1011.5	1.5	1.2
256	4	8	600.5	1008.3	1.5	1.2
512	2	8	596.4	978.4	1.5	1.2
2 потока, размер 2048 МБ						
1	1024	1	515.3	1712.9	1.3	2.1
2	512	1	527.3	1677.3	1.3	2.1
4	256	1	527.6	1747.2	1.3	2.1
8	128	1	526.9	1692.3	1.3	2.1
16	64	1	521.4	1757.2	1.3	2.2
32	32	1	537.5	1716.8	1.4	2.1
64	16	1	552.2	1764.4	1.4	2.2
128	8	1	549.7	1852.1	1.4	2.3
128	8	2	664.1	1905.6	1.7	2.3
128	8	4	801.1	1404.3	2.0	1.7

128	8	8	1074.1	2203.1	2.7	2.7
256	4	8	1088.1	2165.8	2.8	2.7
512	2	8	1102.4	2146.8	2.8	2.6
4 потока, размер 4096 МБ						
128	8	8	1376.5	5432.9	3.5	6.7
256	4	8	1395.3	5356.0	3.5	6.6
512	2	8	1467.7	5209.6	3.7	6.4
8 потоков, размер 8192 МБ						
128	8	8	3158.9	13242.5	8.0	16.3
256	4	8	2762.4	10055.4	7.0	12.4
512	2	8	2781.5	12116.0	7.0	14.9
16 потоков, размер 16384 МБ						
128	8	8	7216.8	13980.0	18.3	17.2
256	4	8	7084.7	13583.5	18.0	16.7
512	2	8	6956.6	13277.0	17.6	16.3
32 потока, размер 32768 МБ						
128	8	8	8516.0	25526.8	21.6	31.4
256	4	8	9160.2	25391.1	23.2	31.2
512	2	8	9885.7	24968.0	25.1	30.7

4. Использование параллельного ввода-вывода в HDF5

4.1 Работа с данными в HDF5

Безусловно, полученные при тестировании модельной версии библиотеки psio результаты интересно сравнить в сходной постановке с результатами, полученными при использовании HDF5.

Напомним, что данные в HDF5 представлены наборами данных (dataset). Каждый набор данных имеет имя, тип данных (скаляр, вектор, массив и пр.), тип элементов (целое, вещественное с указанной точностью). При работе с набором данных в программе создается представление объекта в памяти (dataspace). Набор данных (dataset) в HDF5 может быть многомерным, однако при сохранении этого набора данных в файл он должен быть записан как часть одномерного потока данных. Способ, которым многомерный массив размещается в файле, называется схемой размещения (layout). Самый очевидный способ размещения – сериализация, т.е. преобразование многомерного набора данных в одномерную последовательность байт, записываемую в файл большим монолитным блоком. Эта схема последовательного (contiguous) размещения данных на диске во многом повторяет схему размещения в оперативной памяти.

Альтернативой является кусочная (chunked) схема размещения [6], при которой dataset разделяется на логические куски (чанки), каждый из которых записывается в определенное место внутри файла. При этом чанки могут записываться и считываться в произвольном порядке. Благодаря этим свойствам «кусочной» схемы, её применение позволяет повысить производительность ввода-вывода на ПФС. Ценой этого улучшения обычно становится рост сложности программы из-за необходимости задействовать низкоуровневые возможности HDF5 и учитывать особенности размещения, выравнивания и кэширования данных. В связи с этим, в рамках данной статьи остановимся на последовательной схеме размещения наборов данных.

Доступ к массивам осуществляется через вырезки (hyperslab). Каждая вырезка описывается индексом начального элемента массива и длинам вырезки по измерениям. Размерность вырезки совпадает с размерностью массива. В операциях обмена используется объект memspace, описывающий массив в памяти, который имеет размерность (равную размерности массива или меньше ее) и длины по измерениям. Длины по измерениям memspace должны

совпадать с длинами измерений dataspace. В случае понижения размерности (из массива в операции обмена используется сечение) соответствующая длина по измерению dataset должна быть равна 1.

Общий порядок работы с данными в HDF5 при их создании следующий:

- создание файла,
- создание dataspace для набора данных (описание размерности массива),
- создание memspace, описывающей массив в памяти,
- создание набора данных (присвоение ему имени и выделение пространства в файле),
- определение вырезки на основании dataspace с указанием начального индекса массива в файле,
- выполнение операции записи массива из памяти в файл,
- закрытие набора данных,
- создание следующего набора данных,
- закрытие файла.

Общий порядок работы с данными в HDF5 при их чтении следующий:

- открытие файла,
- открытие набора данных,
- получение dataspace для набора данных из файла,
- создание memspace, описывающей массив в памяти,
- определение вырезки на основании dataspace с указанием начального индекса массива в файле,
- выполнение операции чтения массива из файла в память,
- закрытие набора данных,
- чтение следующего набора данных,
- закрытие файла.

4.2 Конфигурирование параллельного ввода-вывода

В параллельном режиме все функции HDF5 используются как коллективные операции MPI, поэтому процедура создания и открытия файла выполняется специальным образом. На листинге 2 в строках 11-12 показано задание параметров стрипования в HDF5 для ПФС Lustre.

```
 1    MPI_info info;
 2    MPI_Comm comm;
 3 /* Создание атрибутов для открытия файла */
 4
 5    acc_tpl = H5Pcreate(H5P_FILE_ACCESS);
 6    i = MPI_Info_create(&info)
 7    char ss[20], sc[20];
 8    sprintf(ss,"%d", stripe_size);
 9    sprintf(sc,"%d", stripe_count);
10 /* Формирование значений stripe_size и stripe_count */
11    i = MPI_Info_set(info, "striping_unit",   ss)
12    i = MPI_Info_set(info, "striping_factor", sc)
13 /* Настройка ROMIO на коммуникатор задачи и Lustre */
14    ret = H5Pset_fapl_mpio(acc_tpl, comm, info);
15 /* Создание файла с использованием заданных атрибутов */
16     file_id = H5Fcreate(fname, H5F_ACC_TRUNC, H5P_DEFAULT, acc_tpl);
17 /* Закрытие атрибутов файла */
```

```
18    H5Pclose(acc_tpl);
```

Листинг 2. Настройка программы на параллельный ввод-вывод в HDF5
Listing 2. Configuring a program for parallel I/O in HDF5

4.3 Используемая модель данных и реализация

Для максимального приближения к ранее используемой методике оценки производительности ввода-вывода используется следующая модель данных: базовым объектом является двумерный массив M_it (R,BS), где:

- it – номер итерации,
- R – число процессов в задаче,
- BS – длина одиночной порции обмена.

Как и ранее, в подразделе 3.3, с процесса задачи в обмене участвует постоянный объем данных (1 ГБайт). Порция обмена (BS) варьируется от 1 до 512 Мбайт (соответственно используется от 1024 до 2-х итераций).

С процесса задачи в обмене участвует одномерный массив A(BS), представляющий вырезку из массива Mit для строки с номером, равным номеру процесса.

Как и ранее, сначала производится формирование наборов данных (в цикле по итерациям запись с каждого процесса задачи соответствующей строки массива), потом использование наборов данных (в цикле по итерациям на каждом процессе задачи чтение соответствующей строки массива).

На листинге 3 приведен фрагмент исходного текста программы, выполняющей запись набора данных в HDF5.

```
 1      dslen = len * mpi_size;
 2      dimsf[0] = mpi_size;     /* число стрк в файле */
 3      dimsf[1] = len;          /* ширина строки */
 4      dimsm[0] = len;          /* размер обмена */
 5      dataspace = H5Screate_simple(2, dimsf, NULL);
 6      memspace  = H5Screate_simple(1, dimsm, NULL);
 7      sprintf(dsname, "%s-%02d", dsnameproto, it);
 8      dset_id = H5Dcreate(file_id, dsname, dstype, dataspace,
               H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
 9      H5Sclose(dataspace);
10      count [0] = 1;    count [1] = len; /*
                  размеры вырезки по измерениям */
11      offset[0] = rank; offset[1] = 0;   /*
                  начало вырезки по измерениям */
12      dataspace = H5Dget_space(dset_id);
13      status = H5Sselect_hyperslab(dataspace, H5S_SELECT_SET,
                  offset, NULL, count, NULL);
14      data = (int *) malloc(sizeof(int)*dimsm[0]);
15      for (i=0; i < (int)dimsm[0]; i++) {
16          data[i] = rank * 10000 + it * 100 + i%100;
17      }
18      status = H5Dwrite(dset_id, dstype, memspace,
                 dataspace, H5P_DEFAULT, data);
19      fulllen += len * sizeof(int);
20      free(data);
21      H5Dclose(dset_id);
22      H5Sclose(dataspace);
23      H5Sclose(memspace);
24      H5Fflush(file_id, H5F_SCOPE_LOCAL);
```

Листинг 3. Запись набора данных в HDF5
Listing 3. Writing a dataset to HDF5

В строках 1-4 определяются необходимые размерности. В строках 5,6 создаются описание массивов в файле и памяти. В строках 8,9 создается набор данных для хранения массива. В строках 10-13 создается вырезка для работы с подмассивом. В строках 14-17 формируются данные в подмассиве. В строке 18 выполняется запись подмассива в набор данных. В строках 20-23 производится освобождение памяти и созданных объектов. В строке 24 производится сброс файловых кэшей.

4.4 Полученные результаты

В табл. 3 приведены полученные результаты. Методика проведения замеров такая же, как и в разд. 3. Обозначения в табл. 3 совпадают с обозначениями в табл. 1 и 2.

Видно, что параллельная версия HDF5 на данном тесте обеспечивает скорости, сравнимые в полученными ранее на тестах IOR и на структурированных файлах формата PSIO, однако немного уступает им.

Табл. 3. Пропускная способность ПФС на HDF5 на ВК
Table 3. Cluster Lustre bandwith with HDF5 data

BS	Nit	SS	Vw	Vr	Kw	Kr
1 поток, размер 1024 МБ						
1	1024	1	214.0	4827.2	0.5	0.8
2	512	1	266.4	6034.2	0.6	1.0
4	256	1	313.5	6768.3	0.7	1.1
8	128	1	370.9	6887.1	0.8	1.1
16	64	1	421.8	6483.8	0.9	1.1
32	32	1	462.0	5759.2	1.0	1.0
64	16	1	492.0	6012.7	1.1	1.0
128	8	1	544.5	6030.6	1.2	1.0
128	8	2	541.8	6017.6	1.2	1.0
128	8	4	544.0	5909.7	1.2	1.0
128	8	8	542.9	6036.1	1.2	1.0
256	4	8	566.9	5974.9	1.3	1.0
512	2	8	574.9	6057.8	1.3	1.0
2 потока, размер 2048 МБ						
1	1024	1	282.1	836.8	0.6	0.1
2	512	1	273.1	983.3	0.6	0.2
4	256	1	333.5	919.5	0.7	0.2
8	128	1	391.6	1001.6	0.9	0.2
16	64	1	501.0	991.4	1.1	0.2
32	32	1	643.6	1006.8	1.4	0.2
64	16	1	750.5	1053.7	1.7	0.2
128	8	1	812.0	918.0	1.8	0.2
128	8	2	806.6	989.6	1.8	0.2
128	8	4	813.5	1006.5	1.8	0.2
128	8	8	812.1	1171.1	1.8	0.2
256	4	8	885.6	1133.0	2.0	0.2
512	2	8	936.9	1185.5	2.1	0.2
4 потока, размер 4096 МБ						
128	8	8	1422.0	1832.1	3.2	0.3
256	4	8	1591.3	1851.0	3.6	0.3
512	2	8	1671.3	1822.3	3.7	0.3
8 потоков, размер 8192 МБ						
128	8	8	2792.1	2880.0	6.2	0.5
256	4	8	3310.9	3754.3	7.4	0.6
512	2	8	3849.6	4016.1	8.6	0.7

16 потоков, размер 16384 МБ						
128	8	8	3798.7	5773.4	8.5	1.0
256	4	8	4782.3	6136.9	10.7	1.0
512	2	8	5791.6	6470.2	12.9	1.1
32 потока, размер 32768 МБ						
128	8	8	6758.2	14288.6	15.1	2.4
256	4	8	7235.5	14261.0	16.2	2.4
512	2	8	9231.7	14338.9	20.6	2.4

Рассмотрим размещение объектов в файле для рассматриваемой задачи (см. рис. 4).

Рис. 4. Размещение данных в файле формата HDF5
Fig.4. The structure of HDF5 data

Как видно, наборы данных перемежевываются метаинформацией и практически невозможно разместить строки массивов в точных границах сегментов файла. Поэтому при обмене строки массива из процесса задачи с файлом зачастую затрагивается два сегмента файла, расположенных на разных OST, что может приводить к замедлению выполнения операции обмена. Между тем библиотека HDF5 предоставляет приложению возможность запросить выравнивание всех объектов (с размером более некоторого размера) в файле с помощью вызова API функции H5Pset_alignment [7]. Это позволяет выравнивать информацию в файле, подлежащую вводу-выводу, на желаемую границу внутри ПФС. Но такие определенные особенности использования HDF5 совместно с ПФС Lustre являются темой отдельного исследования и в данной статье не рассматриваются.

5. Общие выводы и замечания

В краткой статье невозможно показать все особенности параллельного ввода-вывода для структурированных данных. Приведенные в статье примеры демонстрируют несколько вариантов реализации такого ввода-вывода с использованием стрипованных файлов ПФС Lustre. Для достижения желаемой производительности необходимо учитывать следующее:

- процессы задачи не должны одновременно обращаться к одному и тому же OST;
- в ходе операции обмена со стороны процессов задачи должно задействоваться минимальное число OST;
- блоки данных, участвующие в обмене, должны быть выравнены на границы сегментов файла в ПФС.

Разработанные в ходе исследований модельная версия формата данных PSIO и библиотеки psio позволяют реализовать параллельный доступ к структурированным данным с характеристиками ввода-вывода, сравнимыми с известной в мире библиотекой HDF5.

Авторы надеются, что данная статья окажется полезной разработчикам параллельных прикладных программ, используемых для проведения расчетов на вычислительных системах.

Список литературы / References

[1] HDF Group. Available at: https://www.hdfgroup.org/solutions/hdf5/, accessed 10.05.2023.
[2] Lustre. Available at: https://www.lustre.org, accessed 10.05.2023.
[3] The IO500 benchmark. Available at: http://www.io500.org, accessed 10.05.2023.
[4] Thakur R., Ross R. et al. Users Guide for ROMIO: A High-Performance, Portable MPI-IO Implementaion. Argonne National Laboratory, Technical Memorandum no. 234, 2004, 17 p.

[5] Национальный центр физики и математики. г. Саров / National Center for Physics and Mathematics. Sarov. Available at: https://ncphm.ru/, accessed 10.05.2023 (in Russian).
[6] Chunking in HDF5. Available at: https://support.hdfgroup.org/HDF5/doc/Advanced/Chunking/, accessed 10.05.2023.
[7] Howison M., Koziol Q. et al. Tuning HDF5 for Lustre File System. In Proc. of the Workshop on Interfaces and Abstractions for Scientific Data Storage (IASDS10), 2010, 11 p.

Информация об авторах / Information about authors

Алексей Олегович ИГНАТЬЕВ – начальник лаборатории. Сфера научных интересов: проектирование вычислительных систем, разработка параллельных программ численного моделирования, разработка операционных систем, методы и средства защиты информации.

Alexey Olegovich IGNATYEV – Head of Laboratory. Research interests: design of supercomputer systems, parallel numerical simulation programs development, operating systems development, methods and means of information security.

Сергей Юрьевич МОКШИН – начальник отдела. Сфера научных интересов: проектирование вычислительных систем, разработка функциональных подсистем для высокопроизводительных вычислительных систем, разработка операционных систем, методы и средства защиты информации.

Sergey Yurievich MOKSHIN – Head of Department. Research interests: design of supercomputer systems, development of functional subsystems for high performance supercomputing systems, operating systems development, methods and means for protecting information.

Дмитрий Владимирович ИВАНКОВ – начальник лаборатории. Сфера научных интересов: проектирование многоуровневых систем хранения данных, разработка высокопроизводительных систем хранения данных, исследования методов управления данными.

Dmitry Vladimirovich IVANKOV – Head of Laboratory. Research interests: design of tiered data storage systems, development of high performance storage systems, research in data management methods.

Евгений Александрович БЕКЕТОВ – начальник группы. Сфера научных интересов: проектирование систем хранения данных, разработка средств анализа и управления для систем хранения.

Evgeny Alexandrovich BEKETOV – Head of the work group. Research interests: data storage systems design, analytical methods for storage system management.