004.272.26

Мунерман В.И., Мунерман Д.В.

Смоленский государственный университет (СмолГУ), г. Смоленск, Россия

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ОПЕРАЦИЙ ОБРАБОТКИ ФАЙЛОВ

АННОТАЦИЯ

В статье рассмотрены архитектуры программно-аппаратных комплексов для реализации основных операций в теоретико-множественной (файловой) модели данных. Предложены методы параллельной реализации операций сортировки, выборки, сжатия и слияния строго упорядоченных файлов. Для операции внешней сортировки файла проведена оценка числа процессоров, необходимых для ее эффективной реализации.

КЛЮЧЕВЫЕ СЛОВА

Программно-аппаратные комплексы; модели данных; массовая параллельная обработка данных.

Munerman V.I., Munerman D.V.

Smolensk State University, Smolensk, Russia

THE PARALLEL IMPLEMENTATION OF THE FILES PROCESSING OPERATIONS

ABSTRACT

Architectures software and hardware systems for the implementation of the basic operations in the set-theoretic (file) data model are discussed in the article. Methods of parallel implementation of the operations of sorting, selecting, compression and merging strictly ordered files are offered. For the external file-sorting operation the estimate of the number of processors, which are necessary for its effective implementation, is done.

KEYWORDS

Hardware and software systems; data model; massively parallel processing.

В работе рассматривается возможность создания программно-аппаратных комплексов для параллельной обработки структурированных больших данных. В качестве модели данных, позволяющей строить такие программно-аппаратные комплексы, можно использовать теоретико-множественную или файловую модель (алгебру файлов) [4]. Основной агрегат данных в теоретико-множественной модели – файл, который определяется как фактор-множество множества однотипных записей X по отношению эквивалентности порожденному множеством K и обозначается – X_K. Составляющие X_K классы эквивалентности обозначаются X_{K^*}, или $X_{K^*_{(1)}}, X_{K^*_{(2)}} \ldots$.

Если некоторому экземпляру множества ключей во множестве X не соответствует ни одной реальной записи, считается, что ему соответствует универсальная неопределенная запись Θ. Класс эквивалентности, соответствующий экземпляру множества ключей K^* и состоящий из единственной записи Θ, обозначается Θ_{K^*}.

Каждому файлу в теоретико-множественной модели в реляционной модели данных можно поставить в соответствие отношение во второй или третьей нормальной форме. Операциям над файлами соответствуют операции реляционной алгебры. Это соответствие операций приведено в таблице 1.

Таблица1. Соответствие алгебраических операций в моделях данных

	Алгебра файлов	Реляционная алгебра
1	сортировка	неявная операция, реализована в языке манипулирования данными SQL
2	выборка	selection
3	сжатие	projection
4	слияние строго упорядоченных файлов	теоретико-множественные операции
5	слияние нестрого упорядоченных файлов	join

В рамках теоретико-множественной модели возможны различные эффективные алгоритмы распараллеливания унарных операций сортировки, выборки и сжатия и бинарной операции слияния строго упорядоченных файлов. Параллельная реализация операции слияния нестрого упорядоченных файлов (join) неоднократно рассматривалась авторами [6]. Поэтому в настоящей работе авторы рассматривают проблему создания программно-аппаратных комплексов для первых четырех операций.

1. Сортировка. На вход этой операции подается неупорядоченный по множеству ключей K файл, то есть относительно этого множества ключей его можно рассматривать как множество однотипных записей. Относительно другого множества ключей это может упорядоченный файл. Как правило, объем исходного файла настолько велик, что невозможно ограничиться одним из алгоритмов внутренней сортировки и приходится использовать внешнюю сортировку.

Наиболее популярный алгоритм внешней сортировки, основанный на методе слияния (балансного объединения) легко распараллеливается. На первом этапе этого алгоритма исходный файл делится на равные (за исключением последней) порции, которые сортируются в оперативной памяти параллельно работающих процессоров SP_1, …, SP_M. Для удобства можно считать $M=2^k$. Наименьшее значение M определяется отношением объема файла к объему оперативной памяти процессора, которую можно выделить для внутренней сортировки порции. С другой стороны, время сортировки в основном зависит от величины $\left(\frac{L}{M}\right)^2$ (L – количество записей в файле), что влечет за собой желание увеличить M настолько, насколько позволяет аппаратура. Но после первого этапа выполняется k этапов слияния упорядоченных порций, на каждом из которых число порций уменьшается вдвое, а объем объединенных порций удваивается. Тогда вычислительная сложность рассматриваемого алгоритма будем иметь порядок, который определяется функцией

$$T(L,k)=\left(\frac{L}{2^k}\right)^2+2L\left(1-\frac{1}{2^k}\right).$$

Из этого следует, что даже при значительном увеличении числа записей в файле величина k увеличивается незначительно. Например, при $L=10^5$ $k=17$, а при $L=10^8$ $k=27$. Однако даже при таких небольших значениях k число процессоров, используемых на первом этапе равное 2^k чрезвычайно велико, и значительно превосходит число процессоров (ядер) в современных суперкомпьютерах, и, тем более, в машинах баз данных, и в обычных корпоративных сетях. В некоторой степени выходом может служить использование графических процессоров, количество которых может быть большим и которые весьма эффективны при реализации алгоритмов, подобных внутренней сортировке. Был проведен анализ, суть которого состояла в оценке порядка вычислительной сложности операции сортировки различных значениях k. Результаты анализа для различных значений k приведены в таблицах 2 и 3, а на рисунках 1 и 2 – их графическое представление.

Таблица 2. Оценки сложности параллельного алгоритма внешней сортировки при k =8 и k=10

L	$\min(T(L,k))$	$\frac{\min T(L,k)}{T(L,8)}$	$\frac{\min T(L,k)}{T(L,10)}$
100000	17	2,52	1,09
500000	19	8,63	1,48
1000000	20	16,25	1,95
1500000	21	23,88	2,43
3000000	22	46,77	3,86

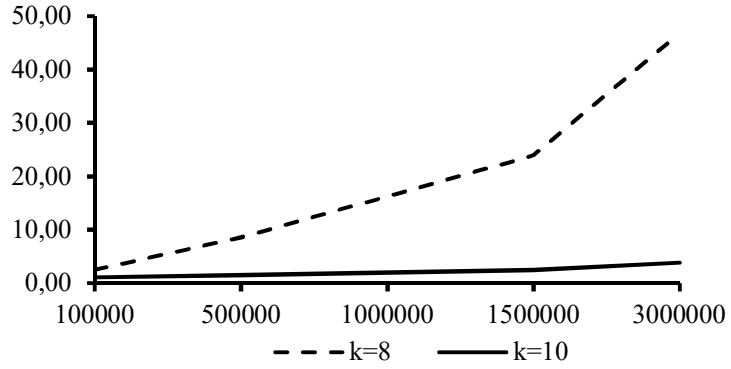

Рис. 1. Оценки сложности параллельного алгоритма внешней сортировки при k =8 и k=10

Таблица 3. Оценки сложности параллельного алгоритма внешней сортировки для k =12, k =14 и k=16

L	$\min T(L,k)$	$\frac{\min T(L,k)}{T(L,12)}$	$\frac{\min T(L,k)}{T(L,14)}$	$\frac{\min T(L,k)}{T(L,16)}$
100000	17	1,01	1,00	1,00
500000	19	1,03	1,00	1,00
1000000	20	1,06	1,00	1,00
1500000	21	1,09	1,01	1,00
3000000	22	1,18	1,01	1,00

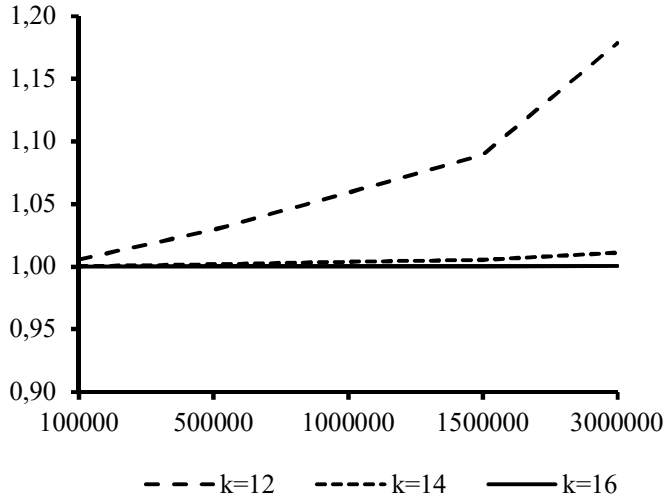

Рис. 2. Оценки сложности параллельного алгоритма внешней сортировки для k =12, k =14 и k=16

Анализ показал, что при достаточно большом количестве процессоров (k =10, 12, 14, 16) отношение $\frac{\min T(L,k)}{T(L,k)}$ стремится к 1. Причем, при значительном увеличении L это отношение мало изменяется. Например, при L=10^8 $\min(T(L,k))$=27, и для k =16 $\frac{\min T(L,k)}{T(L,16)} = 1{,}02$.

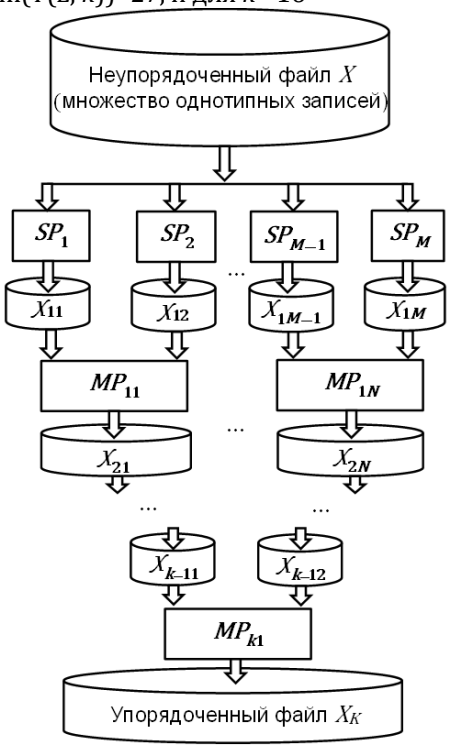

Рис.3. Архитектура программно-аппаратного комплекса для сортировки файла

Программно-аппаратный комплекс для реализации операции внешней сортировки имеет архитектуру, показанную на рисунке 2. На первом этапе $M=2^k$ процессоров (SP_1, …, SP_M) выполняют программу внутренней сортировки файла X, разделенного на порции одинакового размера (за исключением, может быть, последней, если L не кратно M). В результате получается M упорядоченных файлов-порций $X_{11}, X_{12}, \ldots, X_{1M-1}, X_{1M}$. Затем выполняется k этапов, на первом из которых $N=\frac{M}{2}$ процессоров (MP_1, …, MP_N) производят слияние пар, полученных на этапе разделения, в упорядоченные файлы-порции $X_{11}, X_{12} \to X_{21}, \ldots, X_{1M-1}, X_{1M} \to X_{2N}$ удвоенного объема (в записях). На каждом последующем этапе используется вдвое меньше процессоров, а объемы получаемых в результате слияния упорядоченных файлов-порций удваиваются. Перед k-тым этапом остаются два упорядоченных файла-порции $X_{k-1\,1}$ и $X_{k-1\,2}$, объем одного из них равен $\frac{L}{2}$, а объем второго может быть меньше, а в сумме их объемы равны L. На этом этапе используется единственный процессор MP_{k1}, который производит слияния двух последних упорядоченных файлов-порций упорядоченных файлов-порций в упорядоченный файл X_K.

2. Выборка. Самая простая операция обработки файлов, она же и распараллеливается проще остальных операций.

Этой операции соответствует реляционная операция SELECT … WHERE …. Современные языки манипулирования данными в реляционных СУБД содержат средства, обеспечивающие автоматическое распараллеливание этой операции [1, 2]. Здесь предлагается использовать логически-последовательный метод доступа [3]. Этот метод доступа присущ массовой обработке данных, при которой коэффициенты активности (отношение числа обрабатываемых записей к числу записей в файле) всех обрабатываемых файлов близки к единице. Логически последовательным он называется потому, что позволяет учитывать тот факт, что способы организации данных и доступа к ним в различных СУБД могут существенно различаться. Пусть программно-аппаратный комплекс состоит из N процессоров (Select-процессоров). Тогда каждому процессору можно поставить в соответствие фрагмент исходного файла. Объемы этих фрагментов, за исключением, может быть, последнего, равны $\frac{L}{N}$. Как и в случае первого этапа операции сортировки, объем последнего фрагмента может быть меньше этого значения. В большинстве СУБД разделение на фрагменты осуществляется на уровне индексных файлов. При логически-последовательном доступе индексация записей не требуется. Архитектура программно-аппаратного комплекса для операции приведена на рисунке 4. Если файл не разбит на фрагменты физически, каждому процессору ставится в соответствие номер первой записи фрагмента и объем фрагмента в записях. Эту информацию вместе с запросом на выборку Select-процессорам передает хост-процессор. Каждый процессор выполняет выборку из своего фрагмента файла. В случае, когда файл физически разбит на фрагменты, которые находятся на внешних запоминающих устройствах Select-процессоров, хост-процессор передает Select-процессорам только запрос.

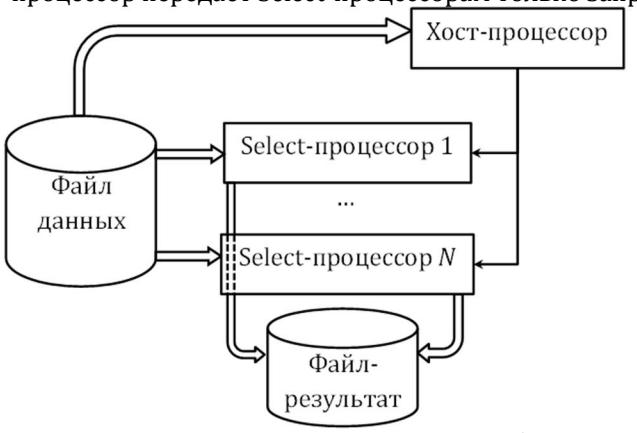

Рис. 4. Архитектура программно-аппаратного комплекса для операции выбоки

3. Сжатие. Пусть даны файлы X_K, нестрого упорядоченный по множеству ключей K, и Y_K, строго упорядоченный по множеству ключей K. Классы эквивалентности этих файлов связаны соотношением $Y_{K^*}=f(X_{K^*})$, где f – функция, реализующая групповую операцию. Тогда считается, что файл Y_K получен из файла X_K в результате применения операции сжатия.

Для реализации этой операции необходимо использовать принцип симметричного горизонтального распределения данных, суть которого состоит в следующем. Исходный файл X_K разбивается на фрагменты, каждый из которых содержит совокупность классов эквивалентности $X_{K^*} \neq \Theta$. Число фрагментов равно числу процессоров в программно-аппаратном комплексе. Эти процессоры называются фрагмент-процессорами. Распределение осуществляется на основе индексно-последовательного метода доступа (ISAM). Для этого файлу X_K ставится в соответствие индексный файл $indX_K = (< K^*_{(1)}, I_{1X}, m_{1X} >, \ldots, < K^*_{(n)}, I_{nX}, m_{nX} >)$. Здесь I_{jX} – индекс первой записи класса эквивалентности $X_{K^*_{(j)}}$, m_{jX} – количество записей в этом классе эквивалентности. Если класс эквивалентности $X_{K^*_{(j)}}$ состоит из единственной универсальной неопределенной записи Θ, то значение I_{jX} не определено, а m_{jX}=0. Естественно, реализация файла $indX_K$ не содержат записи с неопределенными значениями. Для эффективной параллельной реализации операции сжатия необходимо, чтобы объемы фрагментов незначительно отличались. Для того, чтобы это условие выполнялось, в процессе распределения файла используются оптимизационные алгоритмы, такие как множественный алгоритм загрузки рюкзака или предложенный авторами эвристический алгоритм бустрофедона [5]. Эти алгоритмы используют информацию из индексного файла $indX_K$ и распределяют классы эквивалентности файла X_K так, что каждый класс эквивалентности целиком располагается в одном фрагменте, разница объемов фрагментов минимальна, и фрагменты сохраняют упорядоченность исходного файла.

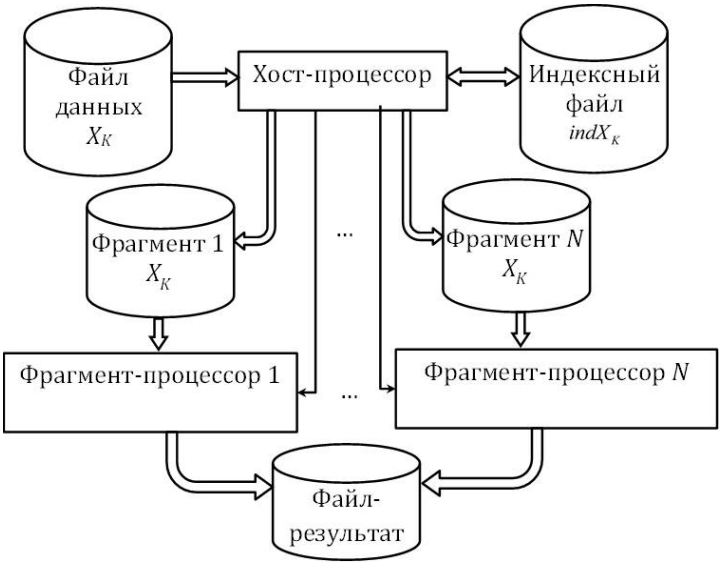

Рис. 5. Программно-аппаратный комплекс для сжатия нестрого упорядоченных файлов

Хост-процессор, если в этом есть необходимость, преобразует исходный файл X_K в N фрагментов. Для этого программным способом формируется индексный файл $indX_K$. На его основе выполняется один из алгоритмов распределения данных, для чего используется один из ранее указанных алгоритмов. Фрагмент-процессоры формируют из каждого класса эквивалентности своего фрагмента файла X_K запись файла результата Y_K.

4. Слияние строго упорядоченных файлов. Пусть даны два файла X_K и Y_K строго упорядоченные по одному и тому же множеству ключей K. В результате слияния этих строго упорядоченных файлов образуется файл Z_K, классы эквивалентности задаются соотношением $Z_{K^*} = f(X_{K^*}, Y_{K^*})$. Функция $f(X_{K^*}, Y_{K^*})$, определенная на классах эквивалентности исходных файлов, задает характер операции.

Параллельная реализация операции слияния строго упорядоченных файлов так же как и операция сжатия требует использования метаданных (индексных файлов). В этом случае один из исходных строго упорядоченных файлов, как правило, имеющий больший объем, определяется как ведущий. Создается индексный файл, определяющий разбиение ведущего файла на равные (за исключением, может быть, последнего) фрагменты. Количество фрагментов равно количеству процессоров. Запись индексного файла для каждого фрагмента содержит адрес первой записи, значения ключей первой и последней записей и объем этого фрагмента. На основе индексного

файла ведущего файла создается индексный файл для ведомого файла. Естественно, фрагменты ведомого файла будут иметь различные объемы, но каждый из них будет содержать записи, значения ключей которых находятся в том же диапазоне, что и значения ключей соответствующего фрагмента ведущего файла. Следует отметить, что некоторые записи ведомого индексного файла будут содержать нулевое значение количества записей. Это означает, что все записи такого фрагмента ведомого файла есть универсальные неопределенные записи Θ, то есть фактически отсутствуют в ведомом файле. Однако в практике массовой обработки данных такие случаи встречаются очень редко. Для удобства эти индексные файлы могут быть объединены в один индексный файл, структура, которого приведена на рисунке 6.

Программно-аппаратный комплекс для слияния строго упорядоченных файлов имеет архитектуру, показанную на рисунке 6. Процессор-коммутатор раздает *N* процессорам, соответствующие записи объединенного индексного файла. Процессоры выполняют программу слияния соответствующих им фрагментов исходных файлов в соответствии с одним из алгоритмов, слияния строго упорядоченных файлов, формируя записи выходного файла.

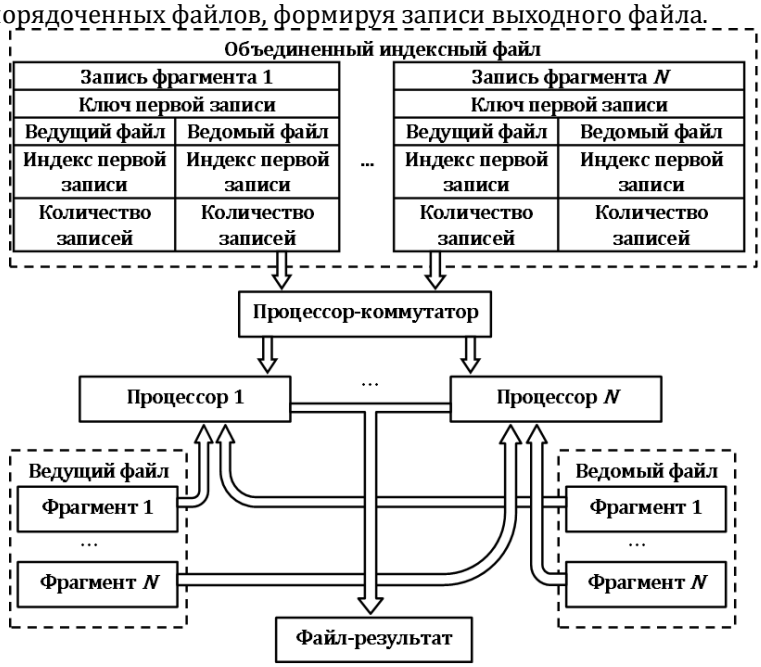

Рис. 6. Программно-аппаратный комплекс для слияния строго упорядоченных файлов

Таким образом, можно утверждать, что для всех операций над файлами, определенными в теоретико-множественной (файловой) модели данных существуют способы распределения данных и основанные на этом распределении параллельные алгоритмы, реализующие эти операции.

Литература

1. Database Data Warehousing Guide. – https://docs.oracle.com/cd/B19306_01/server.102/b14223/usingpe.htm.
2. Exploring SQL Server 2014 SELECT INTO Parallelism. – http://sqlperformance.com/2013/08/t-sql-queries/parallel-select-into.
3. Левин Н.А., Мунерман В.И. Метод логически последовательного доступа к данным. – Системы высокой доступности. 2011. Т. 7. № 4. – с. 65-67.
4. Мунерман В.И. Реализация обработки больших объемов данных на симметричных мультипроцессорных системах. – М.; Системы высокой доступности, 2, 2013, т. 9. – С. 36-39.
5. Мунерман В.И. Опыт массовой обработки данных в облачных системах (на примере WINDOWS AZURE). – Системы высокой доступности. 2014. Т. 10. № 2. – С. 8-13.
6. Мунерман В.И., Мунерман Д.В. Алгебраический подход к построению программно-аппаратных комплексов для повышения эффективности массовой обработки данных. – Современные информационные технологии и ИТ-образование. 2015. Т. 2. № 11. – с. 391-396.

References

1. Database Data Warehousing Guide. – https://docs.oracle.com/cd/B19306_01/server.102/b14223/usingpe.htm.
2. Exploring SQL Server 2014 SELECT INTO Parallelism. – http://sqlperformance.com/2013/08/t-sql-queries/parallel-select-into.
3. Levin N.A., Munerman V.I. Metod logicheski posledovatel'nogo dostupa k dannym. – Sistemy vysokoy dostupnosti. 2011. T. 7. № 4. – s. 65-67.
4. Munerman V.I. Realizatsiya obrabotki bol'shikh ob"emov dannykh na simmetrichnykh mul'tiprotsessornykh sistemakh. – M.; Sistemy vysokoy dostupnosti, 2, 2013, t. 9. – S. 36-39.
5. Munerman V.I. Opyt massovoy obrabotki dannykh v oblachnykh sistemakh (na primere WINDOWS AZURE). – Sistemy

vysokoy dostupnosti. 2014. T. 10. № 2. – S. 8-13.
6. Munerman V.I., Munerman D.V. Algebraicheskiy podkhod k postroeniyu programmno-apparatnykh kompleksov dlya povysheniya effektivnosti massovoy obrabotki dannykh. – Sovremennye informatsionnye tekhnologii i IT-obrazovanie. 2015. T. 2. № 11. – s. 391-396.

Поступила: 12.09.2016

Об авторах:

Мунерман Виктор Иосифович, доцент кафедры информатики Смоленского государственного университета, кандидат технических наук, vimoon@gmail.com;

Мунерман Даниил Викторович, лаборант-стажер кафедры информатики Смоленского государственного университета, danvmoon@gmail.com.