ОРГАНИЗАЦИЯ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ БОЛЬШИХ МАССИВОВ ДАННЫХ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ С МАССОВЫМ ПАРАЛЛЕЛИЗМОМ*

М.Л. Цымблер, Л.Б. Соколинский
Челябинский государственный университет
mzym@csu.ru, sokolinsky@acm.org

Введение

Создание программного обеспечения для обработки больших массивов данных в многопроцессорных вычислительных системах с массивно-параллельной архитектурой является в настоящее время одним из актуальных направлений системного программирования.

Анализ задач, связанных с обработкой больших массивов данных и решаемых на вычислительных системах с массовым параллелизмом (см., например, [2-4]), показывает, что имеется большой класс задач, при распараллеливании которых прикладными программистами используется следующий подход. Вычислительная задача разбивается на подзадачи. Для каждой подзадачи в процессорном массиве системы выделяются непересекающиеся группы процессорных узлов, обычно называемые полями. Каждая подзадача, в свою очередь, разбивается на процессы, которые запускаются на отдельных процессорах в пределах поля, назначенного данной подзадаче.

При реализации задач указанного класса прикладные программисты при разбиении множества процессоров на поля, как правило, не учитывают физическую топологию системы. При этом для реализации межпроцессорных коммуникаций прикладные программисты используют системные сервисы общего назначения (ОС Router для МВС-100/1000), которые часто оказываются недостаточно эффективными для задач с интенсивными обменами данными между процессорами.

В данной работе описывается программный комплекс Омега для распределенной обработки больших массивов данных на отечественной многопроцессорной вычислительной системе МВС-100/1000 [1].

Программный комплекс Омега

Программный комплекс Омега предоставляет системные сервисы, которые могут использоваться прикладными программистами при реализации задач указанного класса. Комплекс рассчитан на использование в массивно-параллельных системах с архитектурой CD_2 [5]. Данная архитектура предполагает разбиение множества процессорных узлов вычислительной системы на непересекающиеся группы процессоров, называемых процессорными кластерами. *Процессорный кластер* состоит из небольшого числа процессорных узлов и имеет фиксированную топологию межпроцессорных соединений. CD_2-система представляет собой множество процессорных кластеров, объединенных высокоскоростной соединительной сетью. На топологию процессорных кластеров накладываются определенные ограничения (например, длина кратчайшего пути между узлами не превышает 2), что позволяет реализовать межпроцессорное взаимодействие более эффективно, чем с помощью сервисов, предоставляемых операционной системой Router для МВС-100/1000.

Программный комплекс Омега предоставляет системные сервисы, позволяющие обрабатывать данные различной природы. В зависимости от объема, времени жизни и частоты обращения к ним, данные могут быть разделены на следующие три категории: персистентные данные, временные данные и сообщения.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 00-07-90077)

Персистентные данные имеют большой объем (не помещаются в оперативную память процессорного узла), характеризуются многократными к ним обращениями и существуют до и после выполнения задачи.

Временные данные имеют большой объем (не помещаются в оперативную память процессорного узла), характеризуются многократными к ним обращениями и существуют только в период выполнения задачи.

Сообщения имеют небольшой объем (помещаются в оперативную память одного процессорного узла), характеризуются однократными к ним обращениями и существуют только в период выполнения задачи.

В соответствии с данной классификацией программный комплекс Омега включает в себя систему передачи сообщений и систему хранения данных.

Система передачи сообщений обеспечивает средства асинхронного обмена данными между любыми двумя процессорными узлами вычислительной системы. В соответствии с принципами CD_2-архитектуры система передачи сообщений состоит из двух подсистем: маршрутизатора и кондуктора. *Маршрутизатор* обеспечивает асинхронную передачу сообщений между узлами, принадлежащими различным процессорным кластерам. *Кондуктор* обеспечивает средства для асинхронной передачи сообщений в пределах одного процессорного кластера. Подсистемы имеют сходный интерфейс, но их реализация использует принципиально различные протоколы обмена, учитывающие топологию процессорных кластеров. Детальное описание данных подсистем приводится в работе [6].

Система хранения данных обеспечивает унифицированный интерфейс обработки персистентных и временных данных на базе следующих устройств хранения: жесткий диск host-компьютера, дисковая подсистема вычислительной системы и электронная дисковая подсистема.

Структура системы хранения

Система хранения данных строится на базе архитектуры клиент-сервер. В соответствии с этим среди процессорных узлов выделяются *узлы-клиенты* и *узлы-серверы*. Серверная часть системы хранения запускается на узле-сервере и обслуживает запросы клиентских частей, запускаемых на узлах-клиентах.

Система хранения предполагает две реализации узла-сервера: аппаратную и программную. *Аппаратная реализация узла-сервера* строится на базе модуля дисковой подсистемы: серверная часть запускается на модуле дисковой подсистемы и использует драйвер дисковой подсистемы. *Программная реализация узла-сервера* строится на базе стандартного процессорного модуля: серверная часть запускается на процессорном модуле, оперативная память которого используется в качестве хранилища данных других процессорных модулей.

Программная структура система хранения данных включает в себя электронную дисковую подсистему и систему управления файлами.

Электронная дисковая подсистема (ЭДП) реализует виртуальный модуль дисковой подсистемы с устройством хранения данных в оперативной памяти процессорного узла и предоставляет соответствующие низкоуровневые сервисы для чтения-записи данных (драйвер ЭДП).

Система управления файлами (СУФ) поддерживает понятие файла, как именованного набора записей фиксированной длины. Файлы используются для унифицированного представления и обработки персистентных и временных данных. СУФ представлена иерархией следующих подсистем: менеджер дисков, менеджер наборов и менеджер файлов. *Менеджер дисков* обеспечивает страничную организацию диска и предоставляет средства для асинхронного чтения и записи указанной страницы диска. Менеджер дисков состоит из клиентской и серверной части. *Менеджер наборов* обеспечивает представление данных, хранящихся на диске, в виде совокупности наборов (связных списков) страниц. Менеджер наборов обеспечивает буферизацию данных на основе единого буферного пула. *Менеджер файлов* обеспечивает представление данных в виде файлов – именованных множеств неструктури-

рованных записей одинаковой длины и обеспечивает стандартные функции для работы с файлами. Подробное описание компонент СУФ и принципы их реализации приведены в работе [7].

Клиент менеджера дисков, менеджер наборов и менеджер файлов составляют клиентскую часть системы хранения данных. Серверную часть системы хранения составляют сервер менеджера дисков и драйвер реальной либо электронной дисковой подсистемы.

Управление буферным пулом

При интенсивных обменах данными между узлами-клиентами и узлами-серверами пропускная способность линков становится узким местом. Данная проблема решается с помощью буферизации данных. *Буферизация* заключается в выделении буферного пула в основной памяти узла-клиента. Если запрашиваемая страница данных уже находится в буфере клиента (ситуация *попадания*), то повторного считывания страницы не происходит, что позволяет значительно сократить объем данных, передаваемых от сервера к клиенту.

При обработке данных большого объема, как правило, не удается обеспечить буферный пул, вмещающий весь массив данных, необходимых задаче. В этом случае приходится *вытеснять* из буферного пула страницы, которые в данный момент не используются. При использовании вытеснения страниц возможна ситуация *неудачи*, когда повторно запрашиваемая страница данных отсутствует в буфере. Отсюда возникает проблема подбора эффективной *стратегии вытеснения страниц*, которая минимизировала бы число неудач.

Известные в настоящее время общие стратегии вытеснения LRU, LFU, FIFO, CLOCK и др. [8] предписывают вытеснять самую "старую" страницу буфера, разными способами определяя "возраст" страниц: по времени последнего обращения к странице, по количеству обращений к странице, по очередности помещения страницы в буфер и т.д. При обработке больших массивов данных общие стратегии часто оказываются неэффективными, например, когда частота обращения к страницам распределена не равномерно. Общие стратегии также могут быть неадекватными при использовании их в системах баз данных, поскольку они не поддерживают избирательное вытеснение страниц [9].

Метод вытеснения страниц, примененный в системе хранения данных программного комплекса Омега, получил название DIR-метода. *DIR-метод* основан на введении статического и динамического рейтингов страниц и использовании избыточного индекса буферного пула. *Статический рейтинг* – это целое число от 0 до 20. Статический рейтинг является атрибутом открытого набора страниц и определяется пользователем при выполнении операции открытия набора. Статический рейтинг страницы остается неизменным, пока набор открыт. При закрытии набора значение статического рейтинга его страниц теряется. *Динамический рейтинг* – это некоторая функция от статического рейтинга, принимающая значения в интервале [0;1[. Значение динамического рейтинга может меняться во время нахождения страницы в буфере. Способ вычисления динамического рейтинга задается системой. *Суммарный рейтинг* страницы получается как сумма статического и динамического рейтингов. Если необходимо освободить место в буферном пуле, то вытесняется страница, имеющая минимальный суммарный рейтинг. Если таких страниц несколько, то вытесняется страница, дольше всего находящаяся в буферном пуле. Механизм статических рейтингов позволяет реализовать избирательное вытеснение страниц. Механизм динамического рейтинга позволяет моделировать различные стратегии вытеснения страниц.

Избыточный индекс буферного пула (DIR) представляет собой таблицу в оперативной памяти, в которой, помимо указателей на образы страниц, находящихся в данный момент в буфере, хранится статистическая информация об использовании этих страниц. Избыточность DIR выражается в том, что после вытеснения страницы из буфера соответствующий элемент в DIR остается. Это позволяет накапливать статистику попаданий и неудач, которая используется для предсказания последующих обращений к страницам диска. Длина DIR определяется как $k*M$, где M – длина буферного пула в страницах, целое $k>1$; k называется кратностью DIR и его точное значение устанавливается экспериментально. При этом $k*M<D$, где

D – общее число страниц на диске (это необходимо для того, чтобы DIR занимал памяти меньше, чем буферный пул).

Примерами основных статистических атрибутов элемента DIR являются счетчик попаданий *HC (hit counter)*, время последнего попадания *HT (hit time)*, счетчик неудач *FC (fault counter)* и время последней неудачи *FT (fault time)*. Указанные статистические атрибуты позволяют моделировать с использованием DIR-метода практически любую общую стратегию вытеснения.

На базе DIR-метода были разработаны несколько оригинальных стратегий, некоторые из которых приведены в Табл. 1. (здесь *NORM* означает функцию нормирования, приводящую значение к диапазону [0;1[).

Табл. 1. Примеры стратегий вытеснения, использующих DIR

Стратегия вытеснения	Подсчет динамического рейтинга страницы
DIR_{FC}	NORM(*HC+FC/k*)
DIR_{FT}	NORM(*HT+FT/k*)

Экспериментальные результаты

Для исследования эффективности предложенного метода буферизации страниц были проведены численные эксперименты. При проведении экспериментов ставились следующие цели:

- сравнить эффективность общих стратегий вытеснения страниц и стратегий, использующих DIR;
- сравнить эффективность различных стратегий, использующих DIR, при одинаковом значении кратности длины DIR;
- определить оптимальное значение кратности длины DIR.

Была разработана программа, моделирующая последовательность *L* случайных обращений к страницам диска 1,…,*N* с Зипфовым (по правилу 80-20) распределением частот обращений (то есть вероятность доступа к странице с номером, не превосходящем *i*, равна $(i/N)^{\log a/\log b}$, где a=0.8 и b=0.2). В качестве *L* и *N* брались значения 3000 и 1000 соответственно.

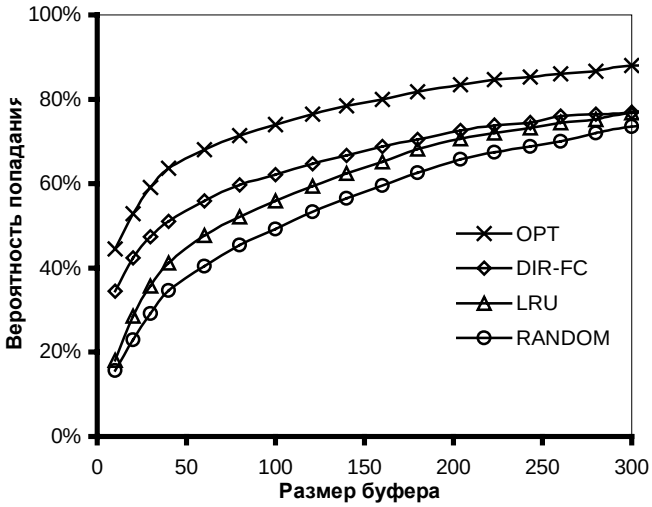

Рис. 1. Сравнение эффективности общих стратегий вытеснения и стратегий, использующих DIR

Для сравнения эффективности были взяты стратегия LRU (которая среди общих стратегий считается лучшей) и стратегия DIR_{FC}. Результаты эксперимента приведены на Рис. 1. Для буферного пула небольшого размера (10-30 страниц) эффективность DIR_{FC} превосходит эффективность LRU на 15%. Затем это превосходство снижается до 10% при размере буферного пула 30-100 страниц и 5% при буфере в 100-200 страниц. При большем размере буфера обе стратегии показывают практически одинаковую эффективность (ожидаемый результат, поскольку при больших размерах буфера вытеснение страниц практически отсутствует).

Для сравнения стратегий, использующих DIR, были взяты две стратегии: DIR_{FC} и DIR_{FT}. Результаты эксперимента приведены на Рис. 2. Эффективность стратегии DIR_{FC} для буферного пула размером 10-60 страниц в среднем выше, чем у DIR_{FT}, на 10%. При большем размере буфера обе стратегии показывают практически одинаковую эффективность.

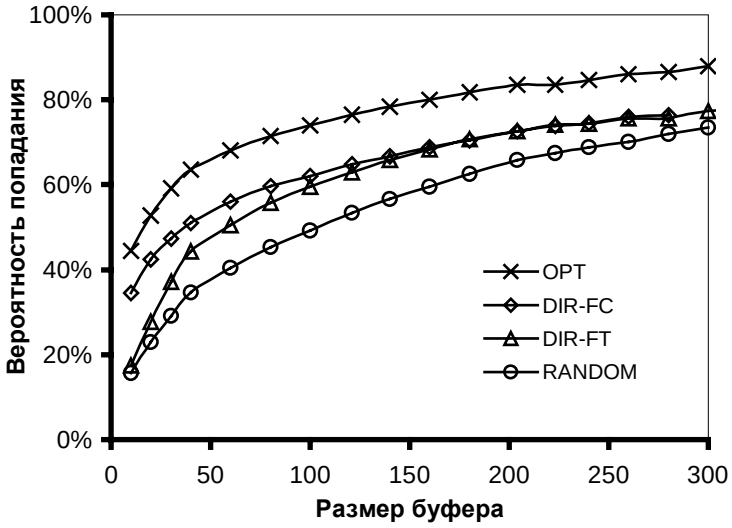

Рис. 2. Сравнение эффективности стратегий вытеснения страниц, использующих DIR

Эксперименты, в которых варьировалось значение кратности длины DIR, показали, что наибольшей эффективности стратегий можно достичь при значении кратности k=0.02*D, где D – общее число страниц на диске.

Заключение

В работе описан комплекс системных программ для распределенной обработки данных в вычислительных системах с массовым параллелизмом. Основными компонентами комплекса являются система передачи сообщений и система хранения данных. Данный комплекс был реализован для отечественной многопроцессорной вычислительной системы МВС-100. Комплекс может быть использован в качестве системного окружения в задачах, требующих интенсивной обработки больших массивов данных.

Предложен оригинальный метод вытеснения страниц из буферного пула, основанный на использовании избыточного индекса буферизованных страниц и введении статических и динамических рейтингов страниц. Приведены результаты численных экспериментов, подтверждающие перспективность данного метода.

ЛИТЕРАТУРА

1. Левин В.К. *Отечественные суперкомпьютеры семейства МВС*. http://parallel.ru/mvs/levin.html
2. Коковихина О.В. *Распараллеливание алгоритма решения задачи о распространении акустических колебаний в газовых потоках* // Алгоритмы и программные средства параллельных вычислений. Сб. науч. тр. Екатеринбург. УрО РАН. 1995. С. 79-85.
3. Мельникова Л.А., Розенберг В.Л. *Численное моделирование динамики блоковой структуры на МВС* // Алгоритмы и программные средства параллельных вычислений. Сб. науч. тр. Екатеринбург. УрО РАН. 1998. С. 221-235.
4. Цепелев И.А., Короткий А.И. и др. *Параллельные алгоритмы решения задачи моделирования высоковязких течений в верхней мантии* // Алгоритмы и программные средства параллельных вычислений. Сб. науч. тр. Екатеринбург. УрО РАН. 1998. С. 301-317.
5. Соколинский Л.Б. *Проектирование и анализ архитектур параллельных машин баз данных с высокой отказоустойчивостью* (см. настоящий сборник).
6. Sokolinsky L.B. *Operating System Support for a Parallel DBMS with an Hierarchical Shared-Nothing Architecture* // Proc. of the 3rd East-European Conf. on Advances in Databases and Information Systems (ADBIS'99), Maribor, Slovenia, September 13-16, 1999. Maribor University Publishing. 1999. P. 38-45.
7. Zymbler M.L., Sokolinsky L.B. *Implementation Principles of File Management System for Omega Parallel DBMS* // Proceedings of the 2nd International Workshop on Computer Science and Information Technologies (CSIT'2000), Ufa, Russia, September 18-23, 2000.
8. Effelsberg W., Harder T. *Principles of Database Buffer Management* // ACM Trans. on Database Systems. Dec. 1984. Vol. 9. No.4. P. 560-595.
9. Stonebraker M. *Operating System Support for Database Management* // Communications of the ACM (CACM). July 1981. Vol. 24. No.7. P. 412-418.