УДК 681.3.06

МЕТОДЫ ОРГАНИЗАЦИИ УПРАВЛЕНИЯ ДАННЫМИ В МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С МАССИВНО-ПАРАЛЛЕЛЬНОЙ АРХИТЕКТУРОЙ

М.Л. Цымблер

Челябинский государственный университет (454021 Челябинск ул. Бр. Кашириных, 129)

(Научный руководитель: канд. физ.-мат. наук, доцент Л.Б. Соколинский)

Аннотация

В работе предлагаются методы построения комплекса системных программ для организации хранения и передачи больших объемов данных в многопроцессорных вычислительных системах с массовым параллелизмом. Для построения программного комплекса используется подход, предполагающий введение в структуре системы управления данными трех уровней абстракции: аппаратного, физического и логического. Основными компонентами программного комплекса являются система хранения данных и система передачи сообщений. Данный комплекс был реализован для отечественной многопроцессорной вычислительной системы МВС.

Введение

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

Важной областью применения массивно-параллельных вычислительных систем являются задачи, требующие обработки больших объемов данных: системы поддержки принятия решений и сверхбольшие базы данных, содержащие графические, картографические, мультимедийные и др. объекты сложной структуры. Одним из наиболее ярких примеров такого рода задач является задача по обработке базы данных EOS/DIS (Earth Observing System/Data Information System) [1] Американского агентства по аэрокосмическим исследованиям (NASA). Как указывается в Асиломарском отчете о направлениях исследований в области баз данных [2], в ближайшем будущем крупные организации будут располагать базами данных объемом в несколько Петабайт. Для эффективной обработки подобных объемов информации требуются параллельные машины с десятками тысяч процессоров.

В настоящее время одной из перспективных отечественных разработок в сфере многопроцессорных вычислительных систем является многопроцессорный вычислительный комплекс МВС-1000 [3]. МВС-100/1000 представляет собой семейство масштабируемых многопроцессорных вычислительных систем с массовым параллелизмом, являющихся совместной разработкой институтов РАН и промышленности.

МВС-1000 используются в ряде академических институтов и университетов России для решения широкого спектра фундаментальных научных и прикладных задач в областях управления динамическими системами и дифференциальных игр, механики сплошной среды, математического программирования, обработки изображений и распознавания образов и др. [4]. Решение многих из указанных задач связано с необходимостью организации обработки больших объемов данных. Однако для МВС-100/1000 в настоящее время отсутствует соответствующее специализированное системное про-

граммное обеспечение, позволяющее хранить и обрабатывать большие массивы данных.

В данной работе предлагается подход к организации системы управления данными в многопроцессорных вычислительных системах с массивно-параллельной архитектурой. В качестве платформы для реализации указанного подхода рассматривается многопроцессорная вычислительная система МВС-100/1000.

1. Аппаратно-программная архитектура МВС

1.1 Аппаратная архитектура МВС

МВС строится по модульному принципу. *Типовой процессорный модуль (ТПМ)* имеет архитектуру с разделяемой памятью и включает в себя вычислительный процессор **CP** и коммуникационный (сетевой) процессор **NP** (см. Рис. 1). Взаимодействие вычислительного и коммуникационного процессоров осуществляется через общую оперативную память (**SRAM**). Кроме этого, коммуникационный процессор имеет собственную приватную память (**DRAM**). Коммуникационный процессор оснащен высокоскоростными внешними каналами (*линками*) для соединения с другими ТПМ и модулями дисковой подсистемы (МДП). МДП представляет собой специальную интерфейсную плату со стандартным коммуникационным процессором, к которой по SCSI интерфейсу могут быть подключены накопители на жестких магнитных дисках. Коммуникационный процессор модуля дисковой подсистемы имеет четыре серийных линка для соединения с ТПМ.

ТПМ устанавливаются в стойках промышленного стандарта, которые могут соединяться между собой для формирования единой вычислительной системы, объединяющей в себе до тысячи процессоров. Управление МВС осуществляется через host-машину, представляющую собой обычный персональный компьютер или рабочую станцию. В качестве вычислительного и коммуникационного процессоров используются изделия зарубежных фирм – Intel, Texas Instruments, DEC-Compaq, Analog Devices. Детальное описание технических параметров ТПМ приводится в работе [3].

В соответствии с классификацией Флинна [5] многопроцессорные системы семейства МВС-100/1000 относятся к классу MIMD. На уровне всей системы МВС можно рассматривать как систему класса MPP, поскольку имеет место распределенность оперативной памяти. На уровне процессорного модуля архитектура МВС подобна архитектуре SMP, поскольку вычислительный и коммуникационный процессоры разделяют общую память. Однако, в отличие от SMP-систем, процессоры, входящие в процессорный модуль МВС, не симметричны. Вычислительный процессор несет нагрузку, связанную с вычислениями, и не занимается обеспечением транзитных передач данных по соединительной сети. Основной задачей коммуникационного процессора, напротив, является организация межпроцессорных обменов данными.

1.2 Операционная система Router

В качестве операционной системы на МВС-100 используется ОС Router разработки ИПМ им. М.В. Келдыша РАН. *ОС Router* [6] представляет собой распределенную операционную систему класса toolset. Она состоит из процессорного сервера, который запускается на каждом процессорном узле и host-сервера (iserver.exe), запускаемого на host-машине. И процессорный сервер, и host-сервер загружаются всякий раз перед загрузкой задачи пользователя.

ОС Router обеспечивает следующие основные функции:

- загрузку программ пользователя с host-машины на процессорные модули;

- обмен данными между программой и host-машиной в виде некоторого подмножества системных функций ввода/вывода UNIX;
- обмен сообщениями по линкам между программами, запущенными на различных вычислительных процессорах.

Задача пользователя может выполняться только на вычислительных процессорах. Задача пользователя представляет собой совокупность процессов. На каждом процессорном модуле может выполняться только один пользовательский процесс. Каждый процесс реализуется в виде программы, которая с точки зрения системы программирования является отдельным загрузочным модулем. Обмен сообщениями возможен между любыми двумя процессами и построен по принципу асинхронных виртуальных каналов. Для обмена сообщениями с другими программами в тексте программы необходимо в явном виде использовать функции ОС Router: "Запустить чтение сообщения от указанного процесса", "Запустить запись сообщения указанному процессу", "Проверить, свободен ли канал для чтения сообщения", "Проверить, свободен ли канал для записи сообщения".

ОС Router не предоставляет средств организации параллельных нитей управления (легковесных процессов) в рамках одного процесса (процессора). ОС Router не обеспечивает средств написания программ на процессорной сети в целом. ОС Router не предоставляет также средств для работы с файлами дисковой подсистемы.

1.3 Иерархический подход к построению систем управления данными

Анализ задач, решаемых на МВС-100/1000, показывает, что имеется большой класс задач, связанных с обработкой больших объемов данных, при распараллеливании которых обычно применяется следующий подход. Вычислительная задача разбивается на подзадачи. Для решения подзадач в процессорном массиве системы выделяются непересекающиеся группы процессорных модулей, называемых *полями*. Подзадача, в свою очередь, разбивается на процессы, и каждый процесс запускается на отдельном процессорном модуле поля. В соответствии со спецификой задач интенсивность межпроцессорных обменов данными внутри поля (в рамках одной подзадачи), как правило, существенно превосходит интенсивность обменов между полями (между подзадачами) вплоть до нескольких порядков.

В качестве примеров задач указанного класса можно привести следующие задачи: численное моделирование высоковязких течений в верхней мантии земной коры [7], численное моделирование динамики блоковой структуры литосферы [8], расчет параметров акустических колебаний в вихревых и газовых потоках [9] и др.

При реализации задач указанного класса прикладные программисты, выполняя разбиение множества процессоров, доступных задаче, на поля, и назначая роли процессоров внутри поля, как правило, не учитывают физическую топологию системы. При этом для реализации межпроцессорных коммуникаций прикладные программисты используют системные сервисы ОС Router. Однако реализация ОС Router рассчитана на произвольную топологию межпроцессорных соединений. Маршрут, по которому сообщение одного процессорного узла передается другому процессорному узлу, недетерминирован и определяется в зависимости от текущей загрузки всех узлов в сети. Вследствие этого функции межпроцессорного обмена, предоставляемые ОС Router, часто оказываются неадекватными и неэффективными для небольших полей с сильносвязной топологией. В связи с этим возникает необходимость в разработке комплекса системных программ, который мог бы использоваться прикладными программистами для эффективного управления данными при реализации задач указанного класса.

Отмеченные особенности задач, связанных с хранением и передачей данных, и рассмотренная выше архитектура МВС-100/1000 делают целесообразным введение в структуре системы управления данными для массивно-параллельных платформ трех уровней абстракции: аппаратного, физического и логического.

Аппаратный уровень представлен несимметричным процессорным модулем, который состоит из коммуникационного и вычислительного процессоров с разделяемой памятью. Обмен данными между коммуникационным и вычислительным процессорами реализуется аппаратно и микропрограммно. Аппаратный уровень отражает особенности реализации процессорного узла.

На *физическом уровне* система представляет собой множество процессорных узлов, объединенных высокоскоростной соединительной сетью. Обмен данными на этом уровне реализуется на основе сервисов, предоставляемых операционной системой типа ОС Router. Физический уровень отражает способ объединения процессорных узлов в единую систему.

На *логическом уровне* множество процессорных узлов системы разбивается на непересекающиеся группы процессоров, называемых кластерами. *Процессорный кластер* состоит из фиксированного числа процессорных узлов, между которыми предполагается интенсивный обмен данными. Обмены данными между узлами разных процессорных кластеров полагаются имеющими относительно низкую интенсивность. Таким образом, на логическом уровне система представляет собой множество процессорных кластеров, объединенных высокоскоростной соединительной сетью. Логический уровень отражает способ разбиения процессорного массива на непересекающиеся кластеры.

2. Методы организации хранения и передачи данных

2.1 Распараллеливание задач обработки данных

Аппаратная архитектура МВС-100 предоставляет два возможных вида реального распараллеливания работ:

1. Распараллеливание путем разделения задачи на подзадачи и выполнение каждой подзадачи на отдельном процессорном модуле в виде *независимого процесса.* Подобный вид параллелизма важен для вычислительных задач, то есть задач, требующих значительных затрат процессорного времени при минимальных обменах данными между задачами.
2. Распараллеливание обменов данными между процессорными модулями, основанное на асинхронном режиме вызова системных функций обмена сообщениями (функций ОС Router). Подобный вид параллелизма важен для задач, связанных с интенсивными обменами данными между процессорными узлами.

Единицей распараллеливания для первого вида обычно является процесс. Единицей распараллеливания для второго вида является *легковесный процесс (нить управления)* [10].

Под процессом понимается задача, запущенная на отдельном процессорном модуле и выполняющаяся на вычислительном процессоре. На одном процессорном модуле не может выполняться более одного процесса. Данное понимание семантики слова процесс совпадает с его трактовкой в ОС Router. В соответствии с этим, обмен сообщениями между процессами базируется на соответствующих операциях ОС Router.

В силу особенностей программно-аппаратной платформы МВС-100/1000 невозможно динамическое перераспределение процессов по процессорам во время работы задачи. Поэтому процессы, нуждающиеся в динамическом планировании, необходимо организовывать в виде нитей (сопрограмм). Пусть, например, у нас имеется два процес-

сора P_1 и P_2 и три процесса ***A***, ***B*** и ***C***. Пусть ***A*** всегда запускается на P_1, ***B*** всегда запускается на P_2, а ***C*** запускается на том из двух процессоров, который в данный момент менее загружен. Для организации подобного планирования создается две задачи ***X*** и ***Y***. ***X*** включает в себя в качестве сопрограмм (нитей) процессы ***A*** и ***C***, ***Y*** включает в себя в качестве сопрограмм процессы ***B*** и ***C***.

Таким образом, для эффективного распараллеливания обменов данными между процессорными модулями в структуре комплекса системных программ для управления данными должна присутствовать *подсистема поддержки легковесных процессов*. Данная подсистема должна располагаться на нижнем уровне системной иерархии и обеспечивать поддержку легковесных процессов (нитей управления), предоставляя средства для диспетчеризации и синхронизации нитей.

2.2 Классификация данных

Распараллеливание задач, решаемых на МВС-100/1000, обычно выполняется путем разделения задачи на подзадачи. Каждая подзадача выполняется на отдельном процессорном модуле в виде независимого процесса.

Данные, обрабатываемые процессом, могут иметь объем, позволяющий целиком поместить их в оперативную память процессорного модуля. Однако на практике часто встречается ситуация, когда данные не могут быть целиком помещены в оперативную память процессорного модуля и обрабатываются по частям.

Введем понятие блока данных как единицы структуризации данных, обрабатываемых процессом. *Блок данных* содержит взаимосвязанные данные, которые не могут обрабатываться процессом по частям (последовательно: сначала одна часть блока, затем другая). Неделимость блока заключается в том, что все его элементы должны быть доступны процессу в течение всего времени обработки блока. Примером разбиения данных на блоки может быть страничная организация файлов базы данных, хранящихся на диске. Примером блока данных в вычислительной задаче является массив коэффициентов системы алгебраических или дифференциальных уравнений.

Мы можем различать блоки данных в зависимости от *количества обращений* процесса к ним во время обработки. Блоки могут быть однократно или повторно (многократно) используемыми. Например, однобайтовое сообщение, передаваемое от одного процесса другому для их взаимной синхронизации является *однократно используемым блоком*, а страница файла базы данных или массив коэффициентов – *повторно используемым блоком*.

Блоки данных могут иметь различное *время жизни*: *до выполнения задачи*, *в период выполнения задачи* и *после выполнения задачи*. В рассматриваемом нами примере сообщение существует только в период выполнения задачи, а страницы файлов базы данных существуют не только в период выполнения задачи, но также до и после выполнения задачи.

Проведенные рассуждения позволяют нам выделить следующие три категории данных, обрабатываемых в вычислительных системах с массовым параллелизмом: персистентные данные, темпоральные данные и сообщения [11].

Персистентные данные – это файлы данных, которые существуют до и после выполнения задачи. Мы будем предполагать, что в общем случае персистентные данные не помещаются в оперативную память процессорного модуля и поэтому обрабатываются блоками. Будем предполагать также, что блоки персистентных данных являются повторно используемыми. Обработка персистентных данных характерна для параллельных систем баз данных, основанных на реляционной или объектно-ориентированной моделях данных.

Темпоральные данные – это данные, которые существуют только в период выполнения задачи. Мы будем предполагать, что темпоральные данные не помещаются в оперативную память процессорного модуля и поэтому обрабатываются блоками. Будем считать также, что блоки темпоральных данных являются повторно используемыми. Обработка темпоральных данных характерна для вычислительных задач большой размерности.

Сообщения – это данные, которые могут быть целиком помещены в оперативную память процессорного модуля. Мы будем предполагать, что сообщения всегда являются однократно используемыми данными и существуют только в период выполнения задачи. Данные такой природы возникают, например, при синхронизации различных процессов.

На основании данной классификации в структуре комплекса системных программ для управления данными в многопроцессорной вычислительной системе с массовым параллелизмом должны выделяться следующие подсистемы: система передачи сообщений и система хранения персистентных и темпоральных данных. *Система передачи сообщений* предоставляет средства обмена сообщениями между любыми двумя узлами вычислительной системы. *Система хранения персистентных и темпоральных данных* предоставляет унифицированные средства для обработки персистентных и темпоральных данных. В реализации функций указанных подсистем используются средства подсистемы поддержки легковесных процессов (нитей управления).

2.3 Методы построения комплекса системных программ для управления данными

Комплекс системных программ для управления данными в вычислительной системе с массовым параллелизмом целесообразно строить из следующих подсистем: менеджер легковесных процессов, система передачи сообщений и система хранения персистентных и темпоральных данных. Общая структура комплекса приведена на Рис. 2.

Менеджер легковесных процессов (нитей) обеспечивает организацию процессов в виде нитей управления и предоставляет средства для диспетчеризации и синхронизации нитей.

Система передачи сообщений обеспечивает средства обмена сообщениями между любыми двумя узлами вычислительной системы. В соответствии с изложенным выше иерархическим подходом к организации системы управления данными, система передачи сообщений строится из двух подсистем: модуля межкластерного обмена сообщениями и модуля внутрикластерного обмена сообщениями.

Модуль межкластерного обмена сообщениями обеспечивает передачу сообщений между узлами, принадлежащими различным процессорным кластерам. *Модуль внутрикластерного обмена сообщениями* обеспечивает средства для передачи сообщений в пределах одного процессорного кластера.

Система хранения персистентных и темпоральных данных обеспечивает унифицированный интерфейс обработки персистентных и темпоральных данных. При этом могут использоваться следующие устройства хранения данных или их комбинации: жесткий диск host-компьютера, модуль дисковой подсистемы и электронная дисковая подсистема. *Электронная дисковая подсистема* реализуется на базе типового процессорного модуля путем использования его оперативной памяти в качестве хранилища данных.

Архитектура системы хранения персистентных и темпоральных данных приведена на Рис. 3. Система хранения данных строится на базе принципов архитектуры клиент-сервер [12].

Процессорные узлы, доступные задаче, разделяются на два непересекающихся подмножества: *узлы-клиенты* и *узлы-серверы*, а в структуре системы хранения выделяются *клиентская часть* и *серверная часть*. На узле-сервере запускается серверная часть системы хранения, которая обслуживает запросы клиентских частей, запускаемых на узлах-клиентах. В качестве узла-клиента выступает типовой процессорный модуль. В качестве узла-сервера может фигурировать как типовой процессорный модуль, так и модуль дисковой подсистемы.

Клиентская часть системы хранения реализуется в виде иерархии двух подсистем: системы управления файлами и драйвера дисковой подсистемы.

Система управления файлами (СУФ) поддерживает представление данных в виде файлов. *Файл* – это последовательный набор записей одинаковой длины. *Запись* состоит из заголовка и одного информационного поля *info*. СУФ предоставляет следующие основные функции:

- создание и удаление файла;
- открытие и закрытие файла;
- выборка, добавление и удаление записей файла;
- организация последовательного просмотра записей файла.

Файл реализуется как набор страниц диска. *Набор страниц* представляет собой связный список страниц диска. Страница состоит из заголовка и одного информационного поля *info*.

СУФ обеспечивает буферизацию страниц диска. *Буферизация* заключается в выделении буферного пула фиксированного размера в оперативной памяти узла-клиента. При выполнении операции чтения страницы набора ее образ помещается в буферный пул. Операция записи страницы набора выполняет модификацию образа этой страницы в буферном пуле. Измененные образы страниц буферного пула сбрасываются на диск в некоторый определяемый системой момент. Поскольку в течение выполнения задачи возможны многократные обращения к одним и тем же страницам файла (набора), буферизация позволяет значительно сократить объем данных, передаваемых от узла-сервера к узлу-клиенту.

Драйвер дисковой подсистемы предоставляет средства для асинхронного чтения и записи страниц диска. Драйвер дисковой подсистемы инкапсулирует аппаратные особенности дисковой подсистемы. Подобный подход позволяет изолировать программный код, зависящий от устройства конкретной дисковой подсистемы, на уровне драйвера дисковой подсистемы, и обеспечивает *аппаратную независимость СУФ*, как компоненты более высокого уровня системной иерархии. Для использования СУФ на другой аппаратной платформе необходимо разработать соответствующий драйвер дисковой подсистемы, а изменения в исходных текстах СУФ не требуются.

Драйвер дисковой подсистемы обеспечивает для СУФ унифицированный интерфейс доступа к данным, находящимся на устройстве хранения. Устройством хранения данных может служить модуль дисковой подсистемы (МДП) или электронная дисковая подсистема (ЭДП). В соответствии с этим возможны две различные реализации драйвера дисковой подсистемы: *драйвер МДП* и *драйвер ЭДП*.

Серверная часть системы хранения запускается на узле-сервере и обрабатывает запросы клиентских частей на чтение-запись данных, хранящихся на дисках.

Сервер системы хранения обслуживает очередь запросов клиентов на чтение и запись страниц диска. Для повышения скорости обработки запросы клиента на чтение и запись разных страниц диска должны выполняться сервером асинхронно, то есть независимо от порядка поступления запросов от клиента. Для поддержания непротиворечивости и целостности данных, хранящихся на диске, запросы клиента на чтение и запись

одной и той же страницы диска должны выполняться сервером синхронно, то есть в порядке поступления запросов от клиента.

Система хранения предполагает *две различных реализации узла-сервера*: на базе модуля дисковой подсистемы и на базе электронной дисковой подсистемы.

В случае реализации узла-сервера *на базе модуля дисковой подсистемы (МДП)* в вычислительную систему встраивается МДП, к которому по SCSI интерфейсу подключаются дисковые накопители. В данном случае серверная часть системы хранения представлена сервером МДП, который запускается на коммуникационном процессоре МДП.

В случае реализации узла-сервера *на базе электронной дисковой подсистемы (ЭДП)* в качестве узла-сервера фигурирует типовой процессорный модуль, оперативная память которого используется в качестве хранилища данных. В данном случае серверная часть системы хранения представлена сервером ЭДП, который запускается на вычислительном процессоре узла-сервера.

Поскольку сервер МДП и сервер ЭДП предполагают запуск на различных аппаратных компонентах вычислительной системы, они должны иметь различную реализацию. В то же время в рамках одной задачи одновременно могут использоваться как серверы МДП, так и серверы ЭДП.

Описанный подход к построению системы управления данными позволяет инкапсулировать особенности используемой аппаратной платформы на уровне драйвера дисковой подсистемы. Данное обстоятельство позволяет существенно сократить доработки при переносе комплекса на сходные аппаратные платформы (например, МВС-1000).

3. Программный комплекс Омега для управления данными в вычислительной системе МВС-100

В соответствии с подходами, описанными выше, нами был разработан комплекс системных программ для управления данными в многопроцессорной вычислительной системе МВС-100, получивший название Омега. Структура комплекса изображена на Рис. 4. Программный комплекс Омега включает в себя менеджер нитей, модуль топологии, систему передачи сообщений и систему хранения данных.

Менеджер нитей обеспечивает поддержку легковесных процессов (нитей управления), позволяющих эффективно выполнять на одном процессорном узле несколько задач в режиме разделения времени. Нити используются в реализации других подсистем комплекса Омега. Детальное описание принципов проектирования и реализации менеджера нитей приводится в работе [13].

Модуль топологии инкапсулирует аппаратные особенности топологии МВС и определяет его как систему, состоящую из фиксированного числа однотипных процессорных кластеров.

Система передачи сообщений обеспечивает средства асинхронного обмена данными между любыми двумя процессорными узлами вычислительной системы. Система передачи сообщений состоит из двух подсистем: кондуктора и маршрутизатора. *Кондуктор* обеспечивает передачу сообщений между узлами одного процессорного кластера. *Маршрутизатор* обеспечивает передачу сообщений между узлами разных процессорных кластеров. Реализация кондуктора и маршрутизатора основана на протоколах обмена сообщениями, предложенными в работе [14].

Система хранения обеспечивает унифицированный интерфейс обработки персистентных и темпоральных данных на базе следующих устройств хранения: жесткий диск host-компьютера, модуль дисковой подсистемы и электронная дисковая подсисте-

ма. Система хранения состоит из двух подсистем: системы управления файлами и электронной дисковой подсистемы.

Система управления файлами (СУФ) поддерживает понятие файла, как именованного набора записей фиксированной длины и предоставляет стандартные функции по работе с файлами. СУФ представлена иерархией следующих подсистем: менеджер наборов и менеджер файлов. *Менеджер наборов* поддерживает представление данных, хранящихся на диске, в виде совокупности наборов (связных списков) страниц и обеспечивает буферизацию данных на основе единого буферного пула. Менеджер наборов использует оригинальный метод управления буферным пулом, описанный в работах [15, 16]. *Менеджер файлов* поддерживает представление данных в виде файлов и обеспечивает стандартные функции для управления файлами: создание, удаление, открытие и закрытие файла, выборка, добавление и удаление записей файла, организация последовательного просмотра записей файла.

Электронная дисковая подсистема (ЭДП) реализует виртуальный модуль дисковой подсистемы на базе типового процессорного модуля, оперативная память которого используется в качестве хранилища данных. ЭДП предоставляет драйвер ЭДП и сервер ЭДП. СУФ использует функции *драйвера ЭДП* для формирования запросов на чтение-запись данных, хранящихся на ЭДП; *сервер ЭДП* обеспечивает обработку этих запросов. Детальное описание методов реализации ЭДП и других подсистем системы хранения приводится в работе [17].

4. Заключение

Предложены методы построения комплекса системных программ для организации хранения и передачи больших объемов данных в многопроцессорных вычислительных системах с массивно-параллельной архитектурой.

Для построения программного комплекса использован подход, предполагающий введение в структуре системы управления данными трех уровней абстракции: аппаратного, физического и логического.

Предложено в зависимости от объема, времени их жизни и частоты обращения к данным, вычислительной системе с массивно-параллельной архитектурой, различать персистентные данные, темпоральные данные и сообщения. В соответствии с этим основными компонентами программного комплекса для управления данными являются система хранения персистентных и темпоральных данных и система передачи сообщений.

Комплекс системных программ для управления данными получил название Омега и был реализован для отечественной многопроцессорной вычислительной системы МВС-100. Подсистемы комплекса Омега инкапсулируют особенности используемой аппаратной платформы, что позволяет минимизировать доработки при переносе комплекса на платформу МВС-1000 и сходные аппаратные платформы. Комплекс Омега был использован в качестве системного окружения в задачах, требующих интенсивной обработки больших массивов данных.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проекты 00-07-90077, 01-07-06038).

ЛИТЕРАТУРА

1. *Dozier J.* Access to Data in NASA's Earth Observing System // Proc. of the 1992 ACM SIGMOD Int. Conf. on Management of Data, San Diego, California, June 2-5, 1992. ACM Press, 1992. P. 1-3.
2. *Бернштейн Ф. и др.* Программа исследований в области баз данных на следующее десятилетие // Открытые системы. 1999. № 1. -С. 61-68.
3. *Забродин А.В., Левин В.К.* Опыт разработки параллельных вычислительных технологий. Создание и развитие семейства МВС // Высокопроизводительные вычисления и их приложения: Труды Всероссийск. науч. конф. (30 октября – 2 ноября 2000 г., г. Черноголовка). М.: Изд.-во МГУ, 2000. С. 3-8.
4. *Сидоров А.Ф., Гасилов В.Л., Кукушкин А.П.* Разработка высокопроизводительных алгоритмических и программных средств на базе параллельных технологий // Алгоритмы и программные средства параллельных вычислений. Сб. науч. тр. – Екатеринбург: УрО РАН, 1995. С. 3-20.
5. *Flynn M.J., Rudd K.W.* Parallel architectures // ACM Computing Surveys. March 1996. Vol. 28. No. 1. P. 67-70.
6. *Лацис А.О.* Разработка ОС коллективного использования для многопроцессорной супер-ЭВМ МВС-100 // Транспьютерные системы и их применение: Тез. докл. Всероссийск. науч. конф. М.: ИПМ им. Келдыша, 1995. С. 17-24.
7. *Короткий А.И., Решетов В.М., Цепелев А.И.* Применение многопроцессорных ЭВМ для моделирования движения вязкой среды // Высокопроизводительные вычисления и их приложения: Труды Всероссийск. науч. конф. (30 октября – 2 ноября 2000 г., г. Черноголовка). М.: Изд.-во МГУ, 2000. С. 265-268.
8. *Желиговский В.А., Пинский В.И., Розенберг В.Л.* Параллельная реализация блоковых моделей динамики литосферы // Распределенные системы: оптимизация и приложения в экономике и науках об окружающей среде (DSO'2000). Сб. докл. Междунар. конф. (Екатеринбург, 30 мая – 2 июня 2000 г.). Екатеринбург: УрО РАН, 2000. С. 315-318.
9. *Коковихина О.В.* Параллельный вариант программы расчета параметров акустических колебаний в вихревых потоках // Алгоритмы и программные средства параллельных вычислений. Сб. науч. тр. Екатеринбург. УрО РАН. 1998. С. 150-162.
10. *Кузнецов С.Д.* Операционные системы для управления базами данных // СУБД. 1996. № 3. С. 95-102.
11. *Цымблер М.Л., Соколинский Л.Б.* Организация обработки больших объемов данных в многопроцессорных системах с массовым параллелизмом // Высокопроизводительные вычисления и их приложения: Труды Всероссийск. науч. конф. (30 октября – 2 ноября 2000 г., г. Черноголовка). М.: Изд.-во МГУ, 2000. С. 186-190.
12. *Orfali R, Harkey D., Edwards J.* Essential Client/Server Survival Guide. NY: John Wiley, 1994. P. 109.
13. *Sokolinsky L.B.* Operating System Support for a Parallel DBMS with an Hierarchical Shared-Nothing Architecture // Advances in Databases and Information Systems, 3rd East European Conf. (ADBIS'99), Maribor, Slovenia, September 13-16, 1999. Proc. of Short Papers. Maribor University Publishing, 1999. P. 38-45.
14. *Sokolinsky L.B.* Interprocessor Communication Support in the Omega Parallel Database System // Proc. of the 1st Int. Workshop on Computer Science and Information Technologies (CSIT'99), Moscow, Russia, January 18-22, 1999. MEPhI Publishing, 1999. Vol. 2. P. 114-123.

15. *Zymbler M.L., Sokolinsky L.B.* Implementation Principles of File Management System for Omega Parallel DBMS // Proc. of the 2nd Int. Workshop on Computer Science and Information Technologies (CSIT'2000), Ufa, Russia, September 18-23, 2000. Ufa State Aviation Technical University, 2000. Vol. 1. P. 173-178.
16. *Цымблер М.Л., Соколинский Л.Б.* Выбор оптимальной стратегии вытеснения страниц в параллельной СУБД Омега для мультипроцессорной системы МВС-100 // Распределенные системы: оптимизация и приложения в экономике и науках об окружающей среде (DSO'2000). Сб. докл. к Междунар. конф. (Екатеринбург, 30 мая – 2 июня 2000 г.). - Екатеринбург: УрО РАН, 2000. - С. 337-340.
17. *Соколинский Л.Б., Цымблер М.Л.* Принципы реализации системы управления файлами в параллельной СУБД Омега для МВС-100 // Вестник Челябинского университета. Серия математика, механика. 1999. № 2(5). С. 176-199.

Научный руководитель Л.Б. Соколинский

Автор М.Л. Цымблер

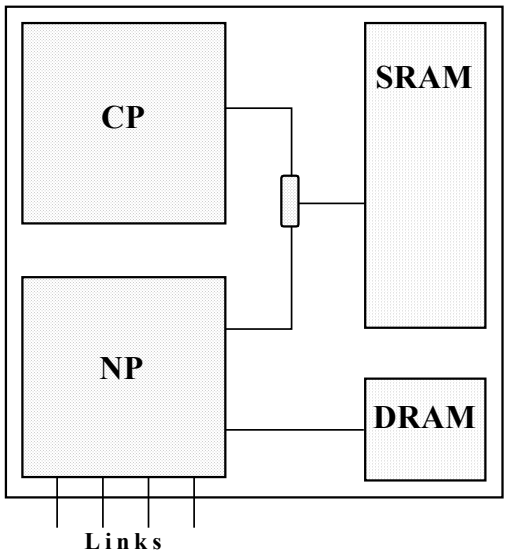


Рис. 1 Структура типового процессорного модуля МВС-100/1000

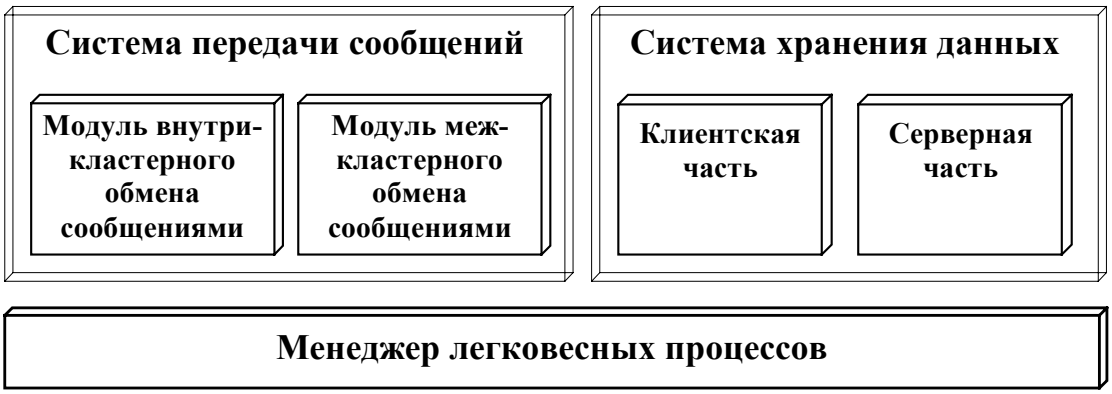
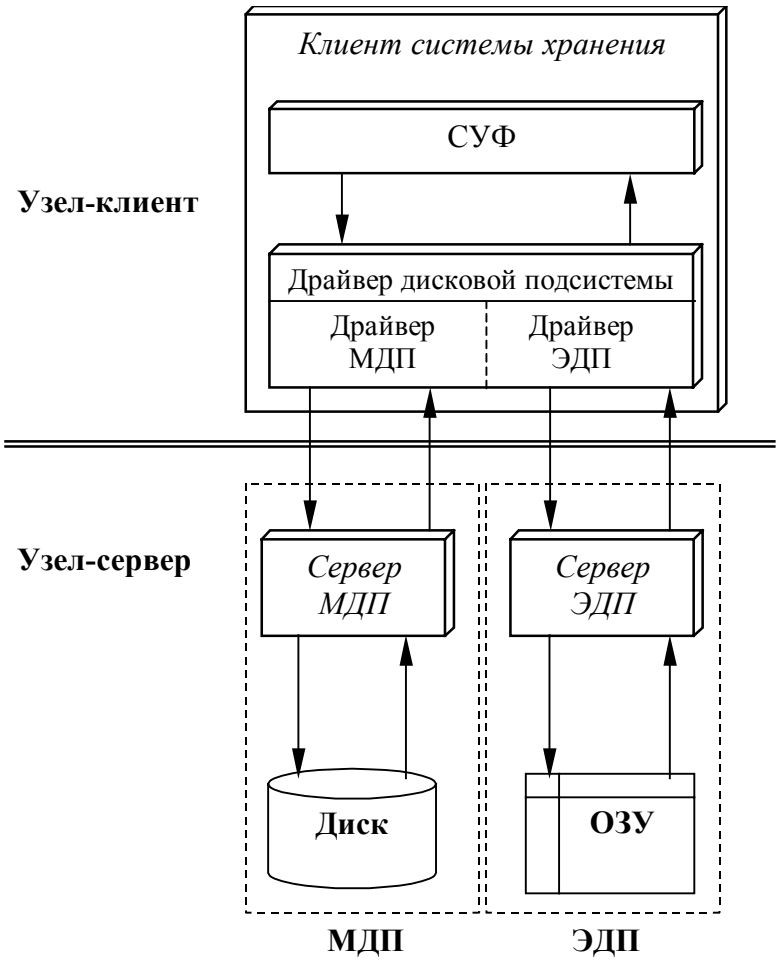


Рис. 2 Структура программного комплекса для управления данными



МДП – модуль дисковой подсистемы
ЭДП – электронная дисковая подсистема

Рис. 3 Архитектура системы хранения персистентных и темпоральных данных

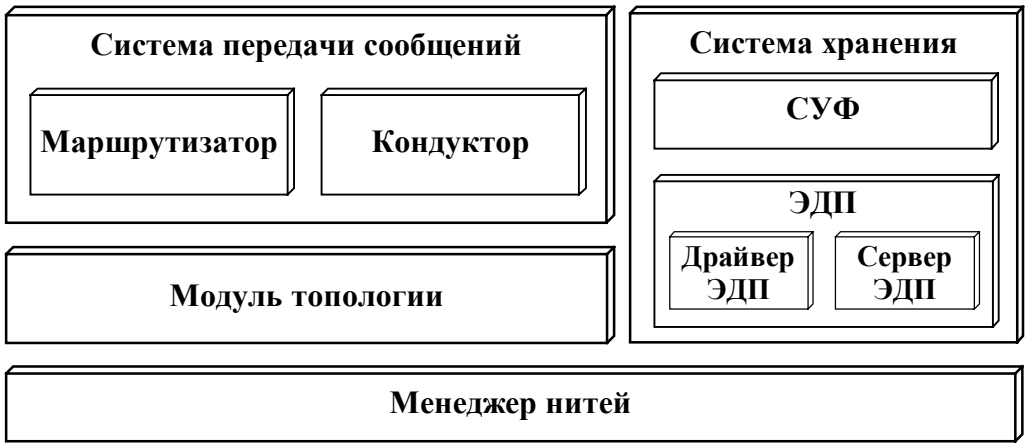


Рис. 4 Структура комплекса Омега