УДК 681.3.06

СРЕДСТВА И МЕТОДЫ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

к.т.н. А.А. Серков, к.т.н. В.Я. Певнев, В.В. Заволодько
(представил д.ф.-м.н., проф. С.В. Смеляков)

В статье рассматривается один из подходов реализации параллельной вычислительной системы, основанный на создании виртуального адресного пространства. Дана сравнительная характеристика алгоритмов программной реализации и конструкторских решений таких систем.

В последнее время все больше внимания стало уделяться параллельным вычислительным системам. Основным параметром классификации параллельных компьютеров является наличие общей (SMP – Symmetric Multiprocessor) или распределенной памяти (MPP – Massive Parallel Processor). Нечто среднее между SMP и MPP представляют собой NUMA - архитектуры (NonUniform Memory Architecture), где память физически распределена, но логически общедоступна [1]. Кластерные (Cluster) системы являются более дешевым вариантом MPP. При поддержке команд обработки векторных данных говорят о векторно-конвейерных процессорах, которые, в свою очередь, могут объединяться в PVP-системы с использованием общей или распределенной памяти. Все большую популярность приобретают идеи комбинирования различных архитектур в одной системе и построения неоднородных систем [2, 3].

Все вышеперечисленные классы суперкомпьютеров имеют один общий недостаток – высокую стоимость, который ограничивает их широкое распространение. Выходом может быть применение кластерной структуры, которая позволяет объединять несколько персональных компьютеров (ПК) посредством высокоскоростного канала связи в один параллельный компьютер.

Для построения кластера необходимо выполнить несколько условий [2]:

1) использование однотипных процессоров (использование процессоров разной мощности затруднено из-за необходимости синхронизации нагрузки);

2) одинаковые операционные системы (применение различных операционных систем в одном кластере невозможно);

3) программы должны быть реализованы с применением технологий MPI (Message Passing Interface) [4] или PVM (Parallel Virtual Machine) [5].

Кроме этого, применяемые методы параллельных вычислений обязывают программиста много времени уделять внимание не только разработке алгоритма, но и передаче данных и синхронизации процессов.

Возможен вариант построения параллельного суперкомпьютера, который может стать наиболее приемлемым для большинства пользователей –

объединение нескольких ПК посредством DSM (Distributed Shared Memory). DSM – это виртуальное адресное пространство, разделяемое всеми узлами (процессорами) распределенной системы. В этом случае возможно объединение как процессоров разной мощности, так и разных операционных систем. Программы в этом случае получают доступ к данным в DSM примерно так же, как они работают с данными в виртуальной памяти традиционных ЭВМ. В системах с DSM данные перемещаются между локальными памятями разных компьютеров аналогично тому, как они перемещаются между оперативной и внешней памятью одного компьютера. Среди основных достоинств DSM (по сравнению с MPI) можно выделить следующие.

В модели передачи сообщений программист обеспечивает доступ к разделяемым данным посредством явных операций посылки и приема сообщений. При этом приходится квантовать алгоритм, обеспечивать своевременную смену информации в буферах, преобразовывать индексы массивов. Все это сильно усложняет программирование и отладку. DSM скрывает от программиста пересылку данных и обеспечивает ему абстракцию разделяемой памяти, к использованию которой он уже привык на мультипроцессорах. Программирование и отладка с использованием DSM гораздо проще. Единственным неудобством применения DSM является необходимость предусматривать, что обращение к памяти может быть не мгновенным, например, когда данный участок памяти находится на другом компьютере или заблокирован.

В модели передачи сообщений данные перемещаются между двумя различными адресными пространствами. Это делает очень трудным передачу сложных структур данных между процессами. Более того, передача данных по ссылке и передача структур данных, содержащих указатели, является в общем случае делом сложным и дорогостоящим. DSM же позволяет передавать данные по ссылке, что упрощает разработку распределенных приложений.

Объем суммарной физической памяти всех узлов может быть огромным. Эта огромная память становится доступна приложению без издержек, связанных в традиционных системах с дисковыми обменами. Это достоинство становится все весомее в связи с тем, что скорости процессоров растут быстрее скоростей памяти и появляются очень быстрые коммуникации. DSM-системы могут наращиваться практически беспредельно и динамически в отличие от систем с разделяемой памятью, т.е. являются масштабируемыми. Программы, написанные для мультипроцессоров с общей памятью (SMP), могут в принципе без каких-либо изменений выполняться на DSM-системах (по крайней мере, они могут быть легко перенесены на DSM-системы). По существу, DSM-системы преодолевают архитектурные ограничения мультипроцессоров и сокращают усилия, необходимые для написания программ для распределенных систем. Обычно они реализуются программно-аппаратными средствами, но в последние годы появилось несколько коммерческих MPP с DSM, реализованной аппаратно (Convex SPP, KSR1).

Алгоритмы реализации DSM. При реализации DSM центральными являются следующие вопросы [4]:

1) как поддерживать информацию о расположении удаленных данных;

2) как снизить при доступе к удаленным данным коммуникационные задержки и большие накладные расходы, связанные с выполнением коммуникационных протоколов;

3) как сделать разделяемые данные доступными одновременно на нескольких узлах для того, чтобы повысить производительность системы.

Рассмотрим четыре основных алгоритма реализации DSM.

Алгоритм с центральным сервером. Все разделяемые данные поддерживает центральный сервер. Он возвращает данные клиентам по их запросам на чтение, по запросам на запись он корректирует данные и посылает клиентам в ответ квитанции. Клиенты могут использовать тайм-аут для посылки повторных запросов при отсутствии ответа сервера. Дубликаты запросов на запись могут распознаваться путем нумерации запросов. Если несколько повторных обращений к серверу остались без ответа, приложение получит отрицательный код ответа (это обеспечит клиент).

Алгоритм прост в реализации, но сервер может стать узким местом.

Этот недостаток устраняется распределением данных между несколькими серверами. В этом случае клиент должен уметь определять, к какому серверу надо обращаться при доступе к разделяемой переменной. Посылка запросов сразу всем серверам нежелательна, так как не снижает нагрузку на серверы. Лучшее решение - распределить данные в зависимости от их адресов и использовать функцию отображения для определения нужного сервера.

Миграционный алгоритм. В отличие от предыдущего алгоритма, когда запрос к данным направлялся в место их расположения, в этом алгоритме меняется расположение данных - они перемещаются в требуемое место. Это позволяет последовательные обращения к данным осуществлять локально. Миграционный алгоритм позволяет обращаться к одному элементу данных в любой момент времени только одному узлу. Обычно мигрируют целиком страницы или блоки данных, а не запрашиваемые единицы данных. Это позволяет воспользоваться присущей приложениям локальностью доступа к данным для снижения стоимости миграции. Однако, такой подход приводит к трэшингу, когда страницы очень часто мигрируют между узлами при малом количестве обслуживаемых запросов. Некоторые системы позволяют задать время, в течение которого страница насильно удерживается в узле для того, чтобы успеть выполнить несколько обращений к ней до ее миграции.

Миграционный алгоритм позволяет интегрировать DSM с виртуальной памятью, обеспечивающейся операционной системой в отдельных узлах. Если размер страницы DSM совпадает с размером страницы виртуальной памяти (или кратен ей), то можно обращаться к разделяемой памяти обычными машинными командами, воспользовавшись аппаратными средствами проверки наличия в оперативной памяти требуемой страницы и замены виртуального адреса на физический. При этом несколько процессов в одном узле мо-

гут разделять одну и ту же страницу. Для определения места расположения блоков данных миграционный алгоритм может использовать сервер, отслеживающий перемещения блоков, либо воспользоваться механизмом подсказок в каждом узле. Возможна и широковещательная рассылка запросов.

Алгоритм размножения для чтения. Предыдущий алгоритм позволял обращаться к разделяемым данным в любой момент времени только процессам в одном узле. Данный алгоритм расширяет миграционный алгоритм механизмом размножения блоков данных, позволяя либо многим узлам иметь возможность одновременного доступа по чтению, либо одному узлу иметь возможность читать и писать данные (протокол многих читателей и одного писателя). Производительность повышается за счет возможности одновременного доступа по чтению, но запись требует серьезных затрат для уничтожения всех устаревших копий блока данных или их коррекции.

При использовании такого алгоритма требуется отслеживать расположение всех блоков данных и их копий. Данный алгоритм может снизить среднюю стоимость доступа по чтению тогда, когда количество чтений значительно превышает количество записей.

Алгоритм полного размножения. Этот алгоритм является расширением предыдущего. Он позволяет многим узлам иметь одновременный доступ к разделяемым данным на чтение и запись. Поскольку много узлов могут писать данные параллельно, то для поддержания согласованности данных требуется контролировать доступ к ним.

Одним из способов обеспечения консистентности данных является использование специального процесса для упорядочивания модификаций памяти. Все узлы, желающие модифицировать разделяемые данные, должны посылать свои модификации этому процессу. Он будет присваивать каждой модификации очередной номер и рассылать его широковещательно вместе с модификацией всем узлам, имеющим копию модифицируемого блока данных. Каждый узел будет осуществлять модификации в порядке возрастания их номеров. Разрыв в номерах полученных модификаций будет означать потерю одной или нескольких модификаций. В этом случае узел может запросить недостающие модификации.

Все перечисленные алгоритмы являются неэффективными. Добиться эффективности можно, изменив семантику обращений к памяти.

Конструкторские решения. Страничная DSM. Общая память разбивается на участки одинаковой длины - страницы или блоки. Если выбрать длину совпадающей (или кратной) длине страницы оперативной памяти процессоров (если их память страничная), то можно будет воспользоваться механизмом защиты памяти для обнаружения отсутствующих страниц DSM и аппаратным механизмом замены виртуального адреса на физический.

К этому же типу DSM (не знающих заранее ничего о содержимом памяти) можно отнести и аппаратные реализации на базе кэшей (Convex SPP).

DSM на базе разделяемых переменных. В DSM системах с разделяемыми переменными только отдельные данные разделяются между процессора-

ми. Программист должен точно определить какие переменные в программе должны разделяться, а какие не должны.

Знание информации о режиме использования разделяемых переменных позволяет воспользоваться более эффективными протоколами когерентности.

DSM на базе объектов. Еще одну группу образуют многопроцессорные системы с объектной организацией распределенной общей памятью. В отличие от всех остальных рассмотренных систем, программы для объектно-ориентированной DSM системы не могут напрямую использовать общие переменные, а только через специальные функции-методы. Система поддержки выполнения параллельных программ, получив запрос на использование некоторой общей переменной, обрабатывает его, поддерживая при этом консистентное состояние разделяемых данных. Весь контроль осуществляется только программными средствами. В тех случаях, когда для балансировки загрузки процессоров применяется миграция данных, воспользоваться соседством расположения данных в локальной памяти процессора затруднительно.

Как известно, самым эффективным путем повышения производительности вычислительных систем является создание параллельных вычислительных структур. Исходя из изложенного выше, можно сформулировать требования к вновь создаваемым системам. Такие системы характеризуются:

- общей разделяемой памятью на программном уровне;
- произвольным составом объединяемых компьютеров;
- возможностью запускать приложения с одного компьютера;
- требуемой надежностью и быстродействием;
- возможностью реализовывать алгоритмы на известных языках программирования, предварительно расширив синтаксис и используя препроцессор.

Для реализации подобной системы необходимо разработать методику, которая, по крайней мере, сочетала бы все преимущества вышеперечисленных методов реализации DSM.

ЛИТЕРАТУРА

1. Крюков В.А. Операционные системы распределенных вычислительных систем. – ИПМ РАН, 1998. – 482 с.

2. Митрофанов В.В., Слуцкин А.И., Ларионов К.А., Эйсымонт Л.К. Разработка и исследование высокопроизводительного SCI-кластера // Сб. научн. тр. – М.: МИФИ. – 2000. – Т. 2. – С. 186 - 187.

3. Забродин А.В., Левин В.К., Корнеев В.В. Массово параллельные системы МВС-100 и МВС-1000 // Сб. научн. тр. – М.: МИФИ. – 2000. – Т. 2. – С. 194 – 195.

4. Distributed operating systems. Andrew S. Tanenbaum, Prentice - Hall, Inc., 1995.

5. Advanced concepts in operating systems. Mukesh Singhal, Niranjan G. Shivaratri, McGraw - Hill, Inc., 1994.

Поступила в редколлегию 08.08.2001