АРХИТЕКТУРА MCVA

В.А. Шляпников

Кластерные системы, хотя и являются сегодня основным классом суперкомпьютеров [8][9][10][11], обладают рядом существенных недостатков, основной из которых: в подавляющем большинстве используются процессоры общего назначения, изначально не рассчитанные на ресурсоемкие задачи. Как следствие, для достижения больших вычислительных мощностей требуется огромное количество таких процессоров (десятки тысяч). Отсюда вытекает второй важный недостаток: плохая масштабируемость из-за сложности координирования огромного количества процессоров. Еще несколько существенных недостатков, вытекающих из огромного количества процессоров, являются большое потребление электроэнергии (порядка 5-10 МВт), большие размеры (требуются огромные помещения), и необходимость специальных систем охлаждения.

Векторные суперкомпьютеры [1][11] также обладают аналогичными недостатками из-за особенностей их организации и структуры. Помимо этого, большая длина векторов делает эти суперкомпьютеры приемлемым решением далеко не всегда.

Изучив текущую ситуацию, автор сделал работы вывод о том, что нужен новый подход к построению суперкомпьютеров. В данной работе рассматриваются три основных аспекта нового подхода: использование векторов малой длины, сокращение управляющих функций в пользу производительности и применение многоядерных процессоров.

Первый аспект нового подхода предусматривает использование векторов малой длины (8-16 элементов). Такие вектора позволят эффективно распараллелить обработку массивов, с очень малым числом остаточных скалярных операций. Также очевидным плюсом является возможность интеграции большего числа векторных регистров непосредственно в ядро. При этом архитектура должна предусматривать возможность обработки всего вектора за один цикл выполнения машинной команды. Именно это и даст возможность многократно увеличить производительность процессоров и, как следствие, суперкомпьютеров. Обмен с памятью может также производиться векторами, что позволяет не только увеличить производительность, но и оптимизировать работу кэш-памяти. Таким образом, частично устраняется основной недостаток кластерных систем: архитектура процессора изначально рассчитывается именно на ресурсоемкие вычисления.

Второй аспект предполагает сокращение управляющих функций в пользу производительности. Каждый процессор вычислительного комплекса должен быть занят вычислениями, а не обработкой прерываний от системных компонентов. Для этой цели могут быть использованы специализированные процессоры и контроллеры (которые могут быть выполнены в виде ПЛИС). Это позволит еще больше сфокусировать вычислительные мощности каждого процессора на выполнении поставленной задачи, оставляя управление системой в стороне. Это же относится к самой организации вычислений: управляющая сторона должна сама готовить задания для вычислительных ядер, освобождая последние для решения других вычислительных задач.

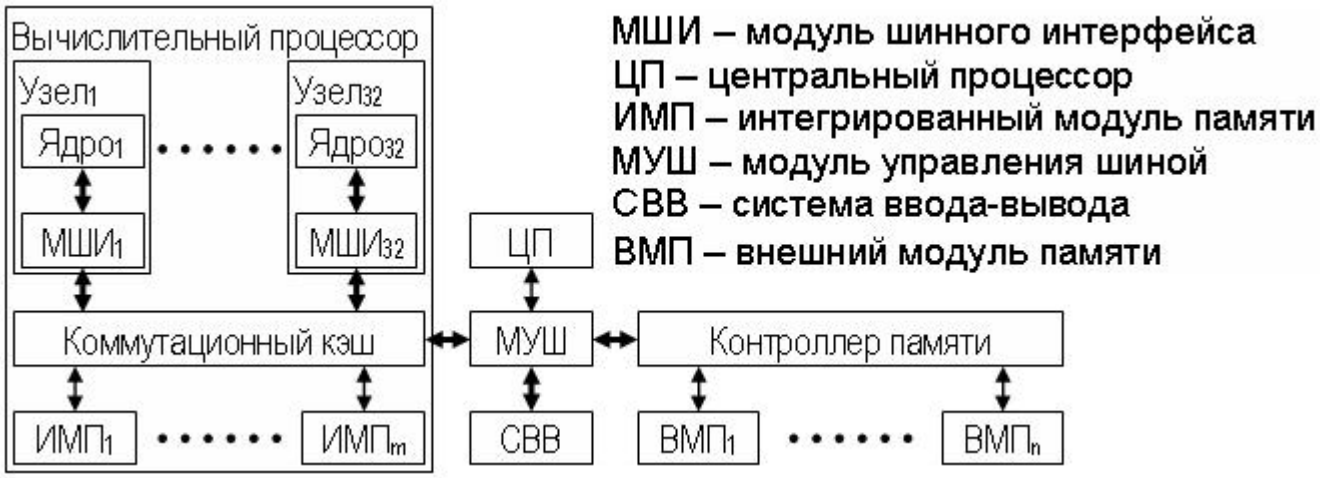

Рис. 1

Третий аспект нового подхода заключается в использовании многоядерных процессоров (с количеством ядер от 16 и выше). Использование таких процессоров позволит не только сократить размеры конечной системы, но и снизить ее энергопотребление. Также возможно встраивание управляющих контроллеров для всех ядер внутри

процессора, тем самим, решая проблему управления системой. Использование многоядерных процессоров также может поднять производительность системы за счет уменьшения различных задержек передачи данных. Развитие полупроводниковых технологий позволит интегрировать все большее число ядер, и поэтому архитектура изначально должна быть рассчитана на многоядерные процессоры.

Основываясь на новом подходе и приведенных аспектах, автор разработал новую архитектуру MCVA (Multi-Core Vector Architecture – многоядерная векторная архитектура). Она предполагает использование 32 вычислительных узлов в одном вычислительном процессоре. Каждый вычислительный узел содержит вычислительное ядро и модуля шинного интерфейса (содержащего модуль управления памятью, и очереди сообщений). Все узлы используют общую память, подключенную через специальный коммутационный кэш. Память может быть интегрирована непосредственно в сам процессор. Вычислительные узлы предназначены для исполнения заданий, инициированных управляющей стороной (которая может быть реализована в виде дополнительного процессорного ядра). Общая схема системы использующей процессор архитектуры MCVA, а также структурная схема самого процессора показана на рисунке 1:

Каждое вычислительное ядро имеет RISC-структуру, и может обрабатывать как скалярные величины, так и 512-битные вектора (могут трактоваться как 16 элементов типа Single, или 8 элементов типа Double). Ядро имеет совмещенный регистровый файл (т.е. скалярные и векторные регистры совмещены), что позволяет избежать дополнительных пересылок (как в традиционных векторных процессорах). Используется только плоская модель памяти, что избавляет от необходимости применения специальных таблиц или сегментных регистров, а также сокращает различные задержки доступа к памяти. Обмен с памятью может быть как скалярными 32-битными значениями, так и векторами, что очень существенно повышает производительность. Каждое ядро может адресовать до 4ГБ памяти. Для увеличения производительности и сокращения числа обращений к памяти используется разделенная кэш-память первого уровня (для данных и команд).

Модуль управления памятью (МУП), входящий в состав модуля шинного интерфейса, преобразует плоскую модель ядра в страничную модель системы. Он содержит встроенную таблицу страничной переадресации (отображается на основную память системы), что позволяет делать преобразование и проверку практически «на лету». Это не скажется на управляемости процессора, поскольку архитектура MCVA предназначена в первую очередь для суперкомпьютеров, где смена контекста большинства вычислительных единиц происходит довольно редко. Поэтому использование встроенной таблицы является большим плюсом архитектуры MCVA, позволяющей гибко конфигурировать процессор (возможно моделирование SMP, NUMA, MPP-систем, а также векторных процессоров). Поддерживается до 1ТБ адресуемой памяти. При возникновении каких-либо ошибок МУП делает запрос на прерывание управляющей стороне, заставляя вычислительное ядро ожидать ее действий. Это сильно упрощает управляющие схемы вычислительного ядра, и позволяет больше внимание уделить именно вычислениям.

Для связи между вычислительным узлом и управляющей стороной используется система коротких сообщений с физической топологией «Звезда», с центром на управляющей стороне. Данная топология обладает большой гибкостью, поскольку на базе этой системы может быть реализована любая логическая топология (в зависимости от политики центра). Также такая организация позволяет делать программную виртуализацию всех вычислительных узлов без существенного усложнения системы. Безопасность также определяется политикой управляющей стороны. Для снижения количества коллизий архитектура предусматривает использование двух очередей (входящих и исходящих) сообщений в каждом вычислительном узле, а также биты свободности и доступности очереди входящих сообщений. Для уведомления ядра используется прерывание, обработчик которого расположен по фиксированному адресу. Используется блокирующий режим работы системы сообщений, т.е. если вычислительное ядро пытается принять сообщение из пустой очереди входящих сообщений, оно будет ожидать поступления сообщения. Если ядро пытается послать сообщение в полную очередь исходящих сообщений, то оно также будет ожидать освобождение этой очереди. Поступление сообщения не вызывает прерывания, если ядро уже обрабатывает одно сообщение, или выполняется попытка принять сообщение из очереди входящих сообщений. Такая схема позволяет реализовать модель «вычисления по запросу» (Calculations On Demand).

Коммутационный кэш является одним из ключевых компонентов процессора разработанной автором архитектуры MCVA, поскольку от его эффективности зависит производительность всего процессора. Поэтому он должен очень быстро обрабатывать запросы всех ядер, читая или записывая требуемые данные из оперативной памяти.

Для конкретной реализации и использования вычислительный процессор может быть реализован в виде платы расширения для некоторой хост-системы (как показано на рисунке 1), которая и будет выступать в роли управляющей стороны. Она должна инициировать необходимые задания для вычислительных узлов (загрузить необходимые данные в память, задать таблицу страничной переадресации и послать сообщение о готовности задания), а также обрабатывать все необходимые запросы вычислительного процессоров. В такой системе может быть установлено несколько вычислительных процессоров, поэтому сама хост-система должна обладать достаточным быстродействием.

Другой вариант реализации, который автор использовал для разработки программного эмулятора, предполагает интегрирование одного или нескольких управляющих ядер в сам процессор, и использование встроенной памяти как основную память системы. Такой вариант менее универсальный, но позволяет добиться большей производительности и меньшего размера за счет сокращения расстояний между управляющей стороной и вычислительным процессором. В качестве архитектуры для управляющего ядра может выступать 32-х или 64-х разрядная архитектура (IA32, PowerPC, MIPS, и т.д.), что позволит различным фирмам выпускать свои реализации процессоров. Это не приведет к хаосу, поскольку архитектура самих вычислительных ядер будет одинакова, а разработчики могут создавать совместимые программные комплексы для различных аппаратных платформ (например, используя технологию Java). Также могут быть разработаны специализированные архитектуры исключительно для использования в составе вычислительного процессора. Именно к таким относится разработанная автором архитектура MRISC. Она имеет некоторые специальные особенности, прекрасно сочетающиеся с особенностями архитектуры MCVA. Описание этой архитектуры не входит в рамки данной работы. Стоит лишь отметить, что она имеет много нововведений, способствующих повышению производительности. На рисунке 2 показана структурная схема такой реализации:

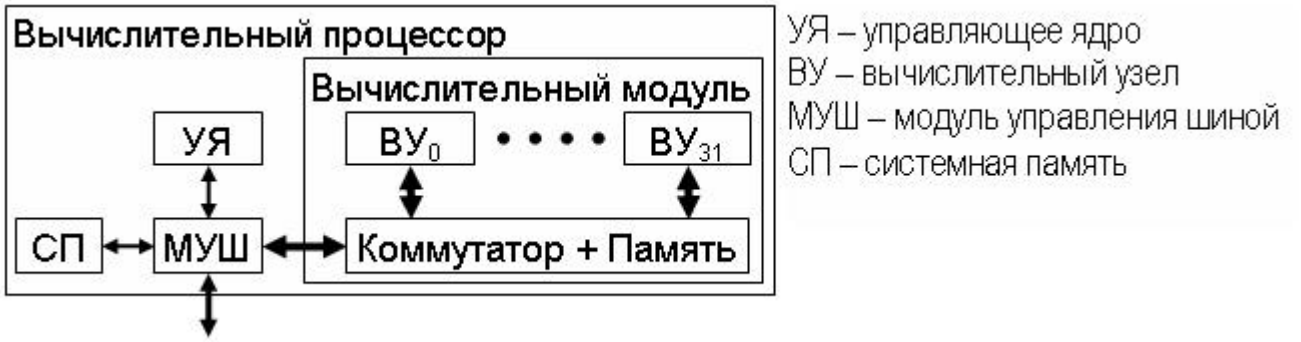

Рис. 2

Основываясь на этой реализации, автор реализовал программный эмулятор процессора MPCEmulator. Использовалась среда программирования Delphi 2007, объектно-ориентированный подход (классы), в сочетании с моделями тактируемых компонентов и подключений к шинам и интерфейсам. Поля классов отражают состав компонента: различные регистры и буферы, а также другие компоненты. Модель тактируемых компонентов предполагает реализацию реакцию на тактовые импульсы в виде методов классов. Действительно, возможно реализовать реакцию на фронт и спад синхроимпульса, которые и будут реализовывать действия процессора (извлечение команды, ее декодирование и исполнение). Модель подключения к шинам и интерфейсам предполагает описание структур этих шин и интерфейсов в виде записей, с последующей передачей указателей на переменные в конструктор класса. Таким образом, компонент как бы «подключается» к шине или интерфейсу. Стоит отметить, что поля классов должны быть объявлены как личные (private), а для нужд визуализации могут быть объявлены публичные (public) свойства, доступные только для чтения (read-only). Такой подход позволит изменять состояние компонента только с помощью определенных методов класса, а также позволит визуализировать компонент без каких-либо затруднений.

Сам проект содержит 21 модуль, где описаны различные компоненты процессора: вычислительное ядро, вычислительный узел, модули памяти, и т.д. Реализация процессора на микропрограммном уровне позволит лучше исследовать возможные пути улучшения новой архитектуры, а также полнее эмулировать работу системы.

Для удобной работы с эмулятором и слежения за компонентами процессора автор разработал визуализирующую среду MCPView. Проект содержит 15 окон, одно из которых является главным. На нем размещены элементы управления тактовыми импульсами, а также позволяет выполнить загрузку или сохранение содержимого системной памяти. Для упрощения начального программирования автором были разработаны компиляторы для простых языков символического кодирования: MCAT для архитектуры MCVA, и MCAsm для архитектуры MRISC. Они доступны из главного окна приложения. Остальные окна визуализируют компоненты и предоставляют доступ к другим окнам. Используется принцип «микросхема под микроскопом», при котором визуализируются все регистры и буферы компонентов, а также состояние шин и интерфейсов. Это позволяет проследить за процессами, происходящими в компоненте, и при необходимости выявить и устранить неисправность.

Итак, в рамках данной работы были выявлены серьезные недостатки современных кластерных систем. На основе их анализа автором был предложен новый подход, а также даны три основных его аспекта. Основываясь на этом подходе, автор разработал новую архитектуру MCVA как основу для построения вычислительных процессоров и суперкомпьютеров.

ЛИТЕРАТУРА:

1. Уильям Столлингс. Структурная организация и архитектура компьютерных систем. Пятое издание. Издательский дом «Вильямс», 2002г. 892 стр.
2. Cell Broadband Engine Architecture – http://www-306.ibm.com/chips/techlib/techlib.nsf/techdocs/1AEEE1270EA2776387257060006E61BA/$file/CBEA_01_pub.pdf
3. Sun UltraSPARC T1 – http://opensparc-t1.sunsource.net/specs/OpenSPARCT1_Micro_Arch.pdf
4. Юров В.И. Assembler. Второе издание. Питер, 2003г. 637 стр.
5. PowerPC User Instruction Set Architecture - ftp://www6.software.ibm.com/software/developer/library/es-archpub1.zip
6. MIPS R4000 Microprocessor User's Manual Second Edition 1994 – http://www.cag.lcs.mit.edu/raw/documents/R4400_Uman_book_Ed2.pdf
7. ARM Architecture Reference Manual – http://www.arm.com/miscPDFs/14128.pdf
8. Курсы интернет-университета "Intuit.ru" - www.intuit.ru
9. Введение в архитектуры многопроцессорных вычислительных систем – http://rsusu1.rnd.runnet.ru/tutor/method/m1/content.html
10. Архитектуры и топологии многопроцессорных вычислительных систем – http://www.informika.ru/text/teach/topolog/content.htm
11. Современные высокопроизводительные компьютеры – http://support.vologda.ru/Book/ARCHITECTURE/Svk/contents.htm