ИМИТАЦИОННАЯ МОДЕЛЬ МНОГОПРОЦЕССОРНОЙ СИСТЕМЫ В СРЕДЕ GPSS WORLD.

Степанова А.С., Четин Е.В.

Проектирование мощных вычислительных систем является сложной технической задачей, одним из аспектов которой является проблема создания системы с заданной производительностью при наименьшем количестве затраченных ресурсов. Обычно используются следующие подходы:

- Эталонные программы, измеряющие производительность (benchmark).
- Тестирование от экспертных организаций.
- Математическое моделирование вычислительной системы. Достоинством данного способа является его гибкость и относительно малая ресурсоемкость.

Производительность вычислительной системы значительно зависит от ее архитектуры. Так, производительность SMP-систем ограничена сверху пропускной способностью системной шины, зато программировать их сравнительно просто, т.к. вся память в системе является общей. Кластерная вычислительная система по мере роста начинает проигрывать в производительности из-за сравнительно низкой пропускной способности межузловых коммуникаций и сложности программно-аппаратных средств.

ccNUMA-системы объединяют в себе как достоинства, так и недостатки SMP и кластера: все пространство памяти в системе имеет единую адресацию, пропускная способность может масштабироваться по мере добавления новых узлов, однако существенной проблемой является поддержка когерентности кэшей. В данной работе моделируется ccNUMA SGI Altix - ccNUMA-система с протоколом кэш-когерентности на основе каталогов. Данные приведены согласно технической документации фирм-производителей [1,2]. Моделируется работа процессорных ядер, кэшей и оперативной памяти, выделяются аспекты архитектуры процессора как элемента вычислительной системы. Рассматривается взаимодействие процессоров с памятью в случае МПС, состоящей из 8 узлов. Описана модель кэш-когерентного протокола, гарантирующего корректность доступа к данным. Модель реализована на языке GPSS World [3].

Узел МПС содержит два двухъядерных процессора, разделяющих общую шину, хаб, выполняющий также функции контроллера памяти и др. В модели одновременно протекают два процесса: выполнение программы в ядре процессора и предзагрузка данных в кэш. При этом другие протекающие в ядре процессы оставлены за рамками этого исследования.

Монтажный блок МПС состоит из 8 узлов и 4 роутеров (Рис.1), соответственно содержит 32 ядра. Каждый хаб соединен с парой роутеров, которые можно моделировать одним двухканальным устройством. Каждый роутер отвечает за адресное пространство 4 лезвий.

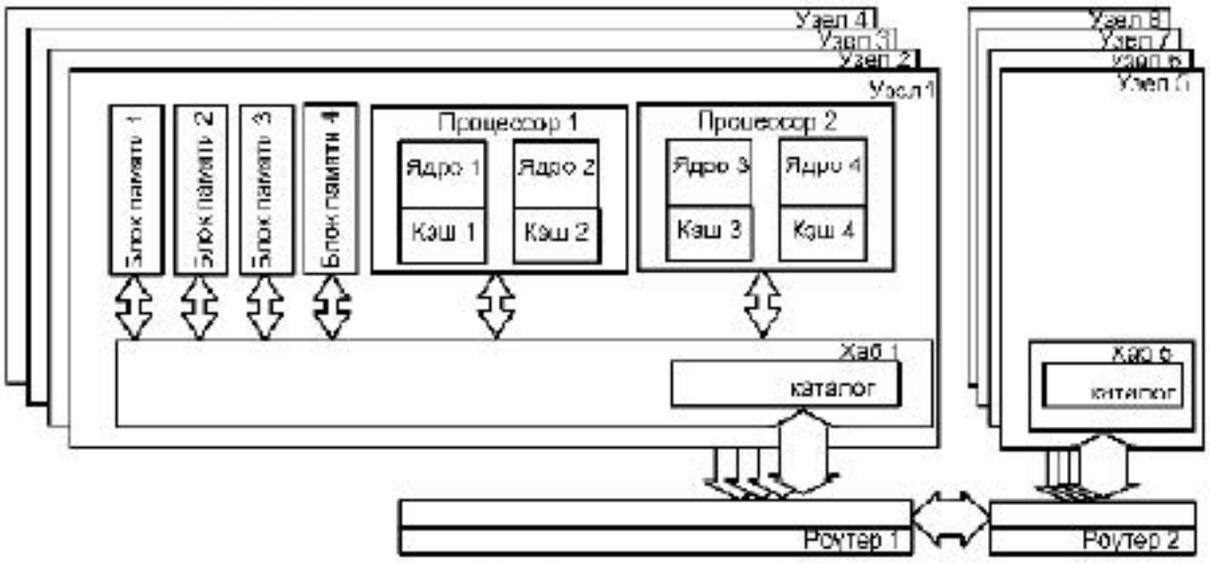

Рис. 1. Схема монтажного блока МПС

В модели используется предположение о программе как о наборе из 1 000 000 полностью параллелизуемых инструкций (заявок в терминах СеМО). На процессоре последовательно генерируется определенное число заявок соответственно решаемой задаче. В случае кэш-промаха процессор обращается к кэшу уровня L1, самому быстрому, работа процессора тормозится. В случае, если данные получены, они отправляются в ядро, и процессор возобновляет работу. Иначе отправляется запрос в кэш L2, и т.д. В случае промаха в кэше L3 формируется запрос к основной памяти. Заявка направляется на хаб, также являющийся контроллером памяти, где она получает координаты исходного блока данных, который может располагаться как в локальной, так и в удаленной памяти. Заявка передается на роутер, отвечающий за исходное ядро, затем на роутер, отвечающий за целевое ядро, потом на хаб, к модулю памяти или кэшу, содержащему целевой блок данных, и обратно, до ядра, сформировавшего заявку, обслуживается на ядре и уничтожается. На обратном пути заявка уже содержит строку данных и испытывает дополнительную задержку из-за конечных пропускных способностей шин. Работа процессора возобновляется после получения данных.

События кэш-попаданий, оптимальное расположение данных (локализация внутри узла), обмен данных между узлами моделируются с помощью генератора случайных чисел: задается вероятность нахождения данных на текущем уровне памяти, затем выбирается случайное число от 0 до 1 и реализуется соответствующее событие. Значения вероятностей задаются параметрически. Соответствующие статистические данные приведены в [4,5].

Параллельно идет предзагрузка данных. Ядро отправляет заявку на хаб, хаб адресует заявку к памяти, память формирует и передает страницу данных с заявкой по обратному пути. Размер страницы может быть от 4 Кб, что соответствует 32 строкам данных. Период генерации запросов предзагрузки обратно пропорционален размеру страницы.

С введением в модель протокола когерентности кэшей запрос к памяти становится более сложным за счет рассылки служебных сообщений другим процессорам, например, совладельцам запрашиваемого блока данных при за-

просе на чтение. Используются несколько упрощенные алгоритмы запросов [6].

В качестве примера применения модели МПС рассматривается рост производительности МПС в зависимости от числа используемых процессоров (рис.2). Наиболее производительны конфигурации из 8, 16, 24 и 32 процессоров. В комбинациях из 9, 17, 25 производительность системы резко падает, т.к. увеличение вычислительной мощности незначительно, зато накладные расходы на обмен данными между узлами возрастают.

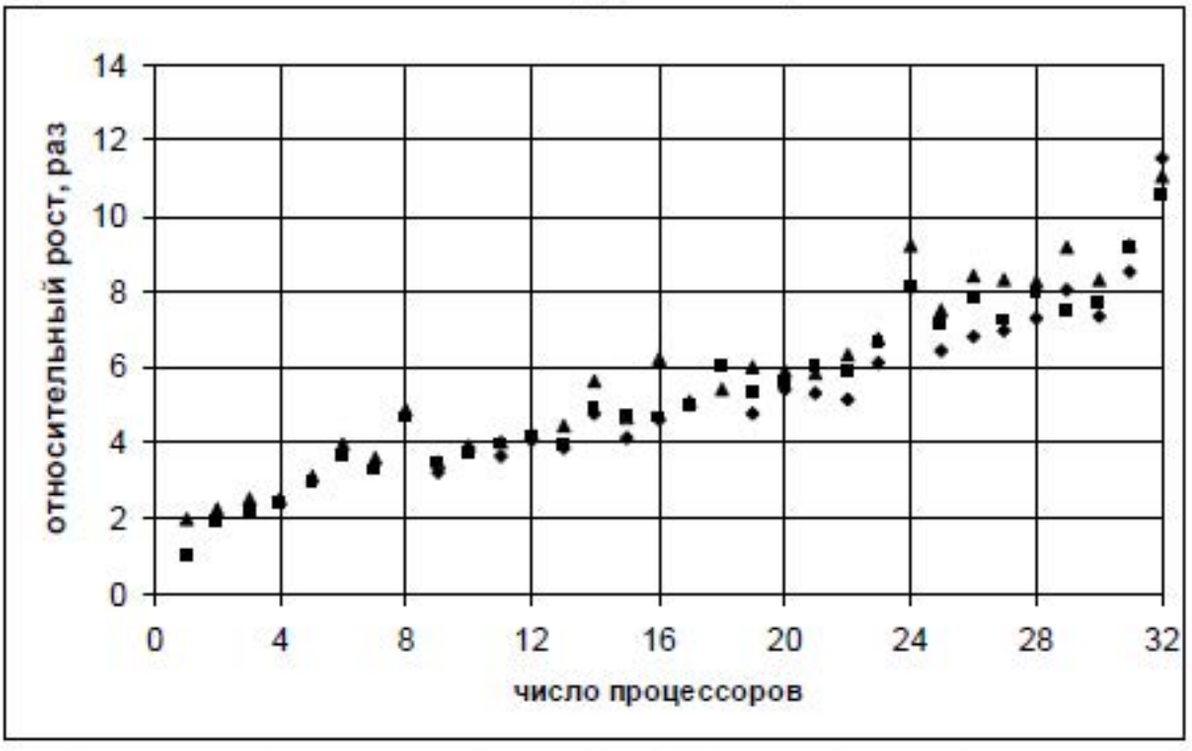

Рис. 2. Относительный рост производительности МПС при увеличении числа процессоров

Литература

1. SGI Altix Applications Development and Optimization // SGI, 2003
2. Dual-Core Update to the Intel Itanium 2 Processor. Reference Manual. For Software Development and Optimization // Intel, 2006
3. Кудрявцев Е. М. GPSS World. Основы имитационного моделирования. --- М.: ДМК Пресс, 2004. --- 320 с.
4. Babka V. Cache Sharing Sensitivity of SPEC CPU2006 Benchmarks. Technical report // Charles University, 2009.
5. Prakash T., Peng L. Performance Characterization of SPEC CPU2006 Benchmarks on Intel Core 2 Duo Processor // 2007 SPEC Benchmark workshop: Proc., January 2007
6. Laudon J., Lenoski D. The SGI Origin: A ccNUMA Highly Scalable Server // ACM SIGARCH Computer Architecture News, 1997.

РАЗРАБОТКА МЕТОДИКИ РАСЧЕТА НАДЕЖНОСТИ ЦЕНТРА ОБРАБОТКИ ДАННЫХ

М.А. Михайлов

На сегодняшний день основным подходом создания корпоративных ИТ-систем, является централизация инфраструктуры. Таким образом, все