ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ МНОГОЯДЕРНЫХ ПРОЦЕССОРОВ В НАУЧНЫХ ВЫЧИСЛЕНИЯХ

М.С. Клинов, С.Ю. Лапшина, П.Н. Телегин, Б.М. Шабанов

При разработке программ параллельной обработки данных, в том числе при использовании автоматических инструментов, необходимо учитывать архитектуру вычислительных систем и имеющийся опыт программирования. Современные процессоры, такие как Intel Xeon 5450 (Harpertown) и Intel Xeon 5675 (Westmere), входящие в состав суперкомпьютера МВС-100К [1], представляют собой четырехъядерный и шестиядерный процессор соответственно, и на материнскую плату могут устанавливаться по два одинаковых процессора. Таким образом, в одном узле с общей памятью могут одновременно работать 8 или 12 ядер.

Такая архитектура похожа на архитектуру классических SMP-систем, однако имеет ряд существенных отличий [2], которые связаны с использованием кэш-памяти двумя процессорами на одной материнской плате и с подключением процессоров. Это делает крайне важным размещение процессов программы по процессорам.

Для использования в научных вычислениях узлы, содержащие многоядерные процессоры, объединяются в кластеры [3], что влечет за собой такие особенности:

- Использование общей кэш-памяти;
- Конфликты по памяти;
- Конфликты по сетевым соединениям;
- Миграция процессов;
- Модели программирования.

Более подробно они описаны в работе [4].

В системах автоматической подготовки параллельных программ, таких как система RATIO [5] и система САПФОР [6] вопрос построения априорной оценки параллельных вычислений для выбора наилучшего варианта распараллеливания программы является крайне острым. При этом важно оценивать варианты распараллеливания с учетом архитектуры вычислительных систем и перечисленных выше особенностей.

При распараллеливании программ без использования инструментов автоматического распараллеливания крайне необходимым является процесс настройки параметров программы, например, расчет и использование оптимальных блоков вычислений и параметров запуска программы. Выполнение такого расчета предлагается существенно автоматизировать.

В работе [4] были описаны формулы для вычисления времени считывания данных из памяти и времени выполнения гнезда циклов. Был рассмотрен случай широко используемой в научных вычислениях векторную операцию $x = x + a*b$.

При использовании многоядерной архитектуры и больших объемах данных требуемые элементы векторов не успевают считываться из памяти, и возникает конкуренция за память. В этом случае время считывания из памяти можно принимать равным [4]:

$$T = \frac{2*N}{V_b*(1 - CR(2*N))} \qquad (1)$$

где

N – длина вектора для каждого процесса ($0 <= i <= N$)

V_b – скорость считывания данных из памяти.

Здесь предполагается, что чтение производится «одним куском», без системных прерываний.

В этой формуле используется коэффициент CR, который определяется для конкретной системы с учетом объема используемых данных. Такой способ позволяет проводить оценки при небольшом использовании вычислительных ресурсов, например, для выбора параметров параллельной программы, в том числе автоматического выбора, с учетом многоядерной архитектуры.

Рассмотрим случай, когда вычисления оформлены в виде гнезда циклов.

```
do i1 = 1, n1
   …
      do ik = 1, nk
          x(i1) = f(i1,…,ik, данные)
       enddo

enddo
```

Время выполнения гнезда циклов на разнородной вычислительной системе при дуплексной передаче данных по коммуникационной сети можно определять по формуле, приведенной в [4]:

$Tц=$

$$\max_{s,t}(\max(T^s_{ц,}T^t_{ц,}) + \max(\sum_s T^{st}_{send}(n^t_s)*(1-CL(s,t,n^t_s)), \sum_t T^{ts}_{send}(n^t_s)*(1-CL(t,s,n^s_t)))) \quad (2)$$

где

$T^s_ц$ - время выполнения части цикла, назначенного на систему *s*:

$T^s_ц = T^s_{org} + L^s/N^s\ (1+CR) + T'_{send}{}^{st}(n^t_s) + T''_{send}{}^{ts}(n^s_t)$,

$T_{send}{}^{st}(X)$ - время передачи *X* байт в из системы *s* в систему *t,*

$T_i{}^s$- время выполнения одной итерации в системе *s,*

n^t_s - - количество передаваемых данных из системы *s* в систему *t,*

N^s – количество используемых процессоров в системе *s,*

L^s - количество итераций цикла в системе *s,*

T^s_{org} – время организации выполнения цикла в системе *s,*

$T'_{send}{}^{st}(X)$ - время инициализации передачи данных из системы *s* в систему *t* (определяется в первую очередь буферизацией),

$T''_{send}{}^{ts}(X)$ - время инициализации передачи приема из системы *t* в систему *s* (определяется в первую очередь буферизацией),

CL – коэффициент конфликтов по памяти, зависящий от системы, объема данных и блокировки коммуникационной сети, аналогично формуле (1).

Для определения оптимальной загрузки требуется минимизировать функцию (2). .

Здесь также предполагается, что выполнение циклов производится без системных прерываний.

Для определения оптимальной загрузки требуется минимизировать функцию (2).

Для определения необходимых для вышеприведенных формул параметров была разработана программа LAPTEST, которая определяет производительность:

1. ядер при выполнении операций с плавающей точкой;
2. оперативной памяти при выполнении операций чтения-записи;
3. коммуникационной среды при выполнении вызовов MPI;
4. подсистемы ввода-вывода при выполнении файловых операций.

Данная программа во время работы обеспечивает выборочную нагрузку на заданные компоненты вычислительной системы.

Программа может в цикле выполнять последовательность действий, которая задается через базовые тесты:

- загрузка процессора (используется функция cblas_dgemm [7]);
- загрузка коммуникационной сети;
- загрузка файловой системы;
- интенсивная работа с оперативной памятью;
- простой.

Программа может работать с любым числом процессов. При этом любому процессу или группе процессов можно задавать индивидуальные задания.

Работа тестов делится на этапы. На каждом этапе можно задавать последовательность тестов для отдельного процесса или группы процессов.

Можно создавать производные тесты, задавая параметры (время, память, загрузка, режим выполнения) индивидуальным тестам. Производные тесты создаются из уже описанных тестов изменением параметров с помощью модификаторов. При тестировании загрузки процессора можно использовать дополнительно до 7 динамических библиотек BLAS. Базовые настройки вынесены в конфигурационный файл.

Кроме того, одним из примеров задач, для которых важна настройка параметров, является задача умножения матриц с разными размерами блока, описанная в [4].

Результаты выполнения такой задачи на процессоре Intel Xeon 5450 (Harpertown) приведены на рис. 1, на котором можно видеть, что после заполнения кэш-памяти происходит снижение производительности до 2 раз.

На рис. 2 и 3 представлены результаты выполнения на одном вычислительном узле при загруженности одного или всех ядер узла соответственно. В каждом случае вычислительный узел имел общую для всех запущенных ядер память, запуски проводились на восьми ядрах системы с использованием процессоров Intel Xeon 5450 (Harpertown) и на двенадцати ядрах с использованием процессоров Intel Xeon 5675 (Westmere).

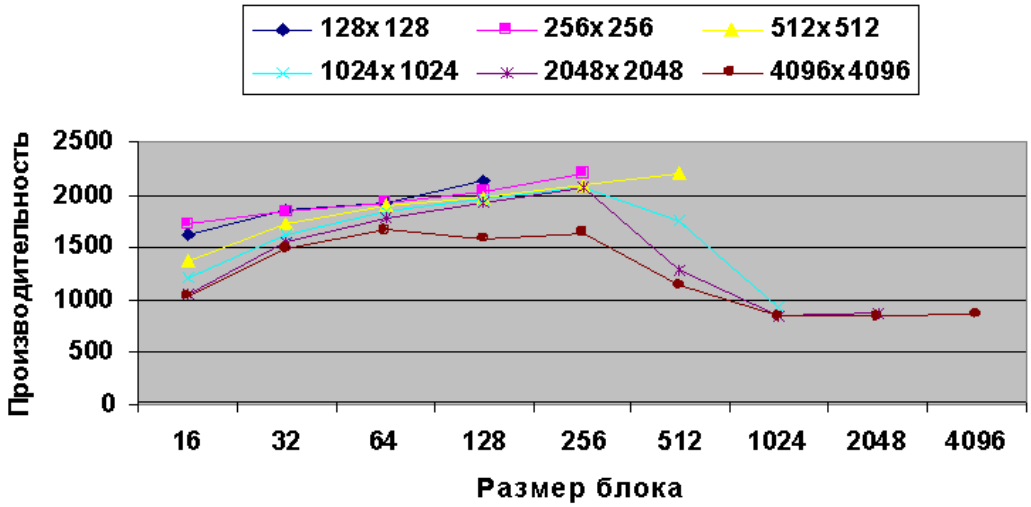

Рис. 1. Блочное умножение матриц. Зависимость производительности (МФлопс) одного ядра от размера матрицы и размера блока. Занято одно ядро процессора Intel Xeon 5450 (Harpertown)

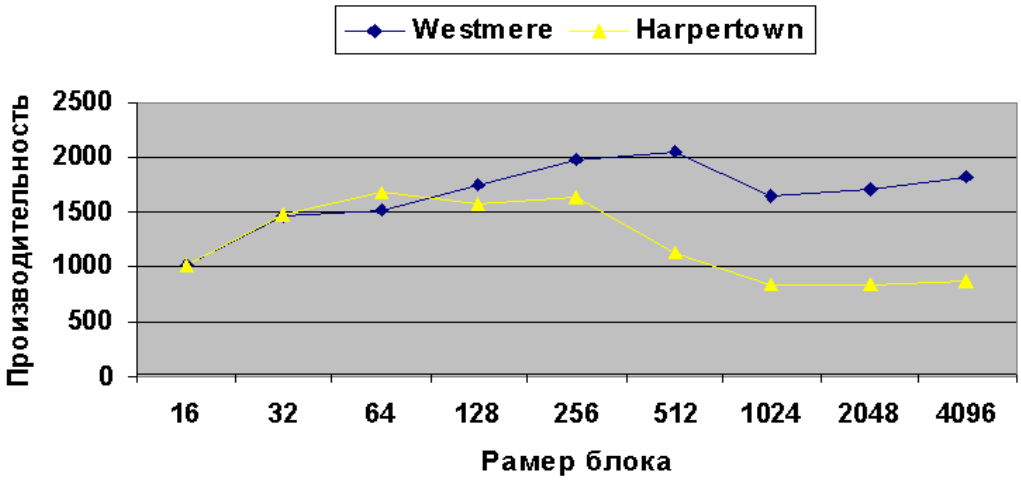

Рис. 2. Блочное умножение матриц 4096x4096. Зависимость производительности (МФлопс) одного ядра от размера блока. Занято одно ядро

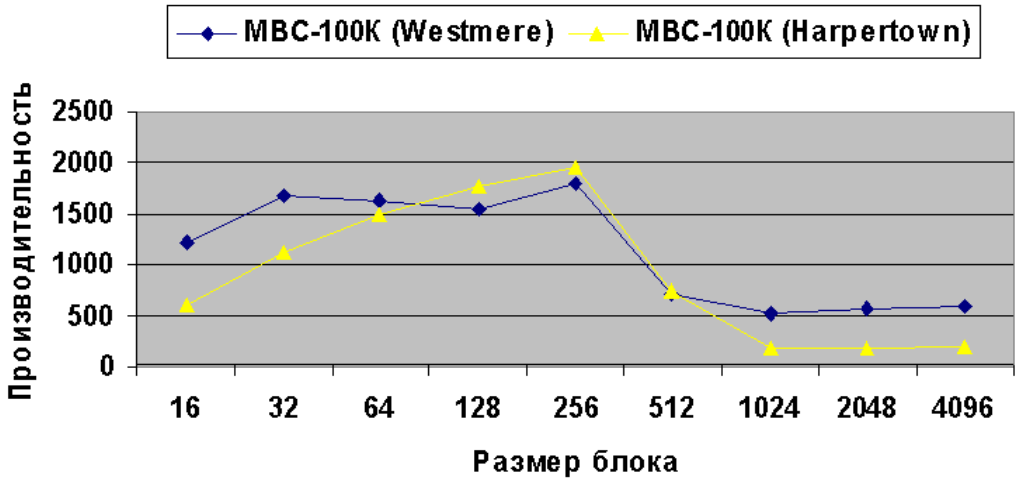

Рис. 3. Блочное умножение матриц 4096x4096. Зависимость производительности (МФлопс) одного ядра от размера блока. Заняты все ядра

Таким образом, для процессора Harpertown CR(16777216) = 3,29 (при 12 ядрах), процессора Westmere CR(16777216) = 2,11(при 8 ядрах).

В качестве примера работы с целыми числами рассмотрим результаты тестирования программы моделирования процессов распространения массовых эпидемий комплекса BIOCLUST[8], разработанного в МСЦ РАН. На рисунке 4 приведены времена работы в зависимости от количества используемых ядер процессора Harpertown на одном узле при выполнении всей программы на 48 ядрах. Коэффициент CR составляет 0,02 при использовнии 2 ядер, 0,04 при использовании 4 ядер и 0,18 при использовании 8 ядер.

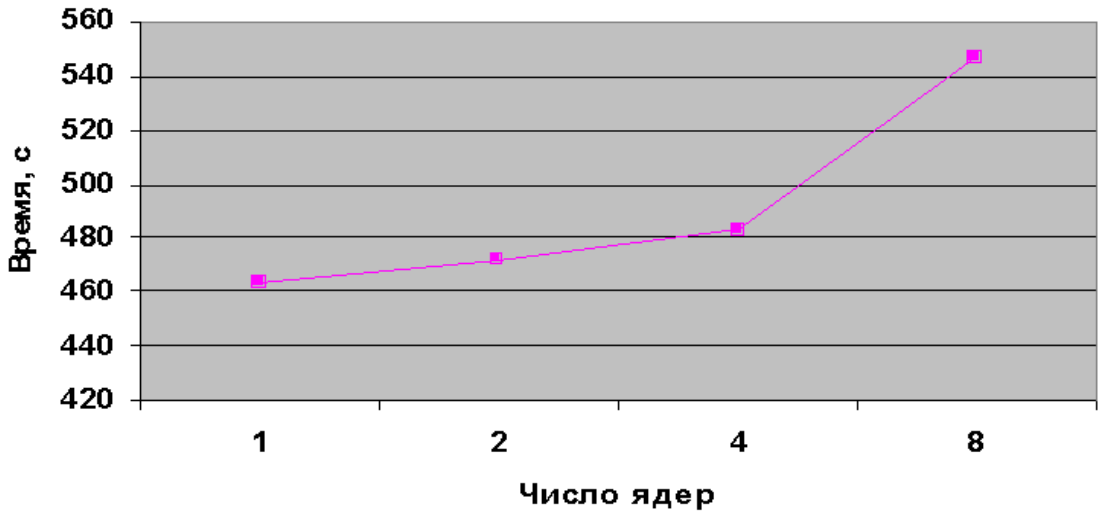

Рис. 4. Зависимость времени выполнения программы комплекса BIOCLUST в зависимости от используемых ядер на процессоре

На рис. 5 представлены результаты выполнения программы на разном количестве ядер системы HP Proliant с использованием процессоров Intel Xeon 5450 (Harpertown).

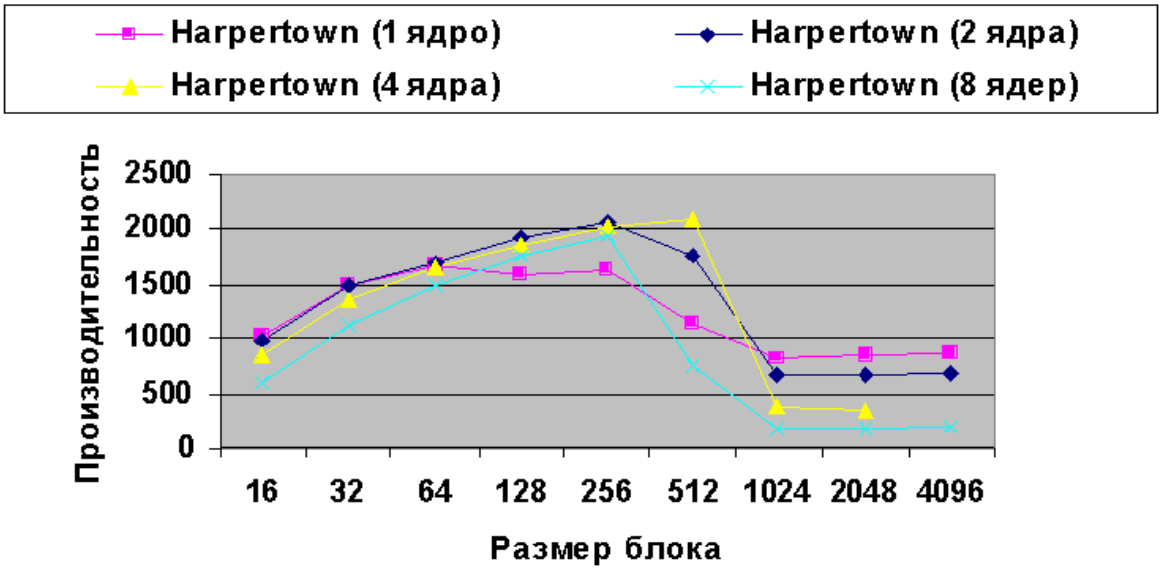

Рис. 5. Блочное умножение матриц 4096x4096. Зависимость производительности (МФлопс) одного ядра от размера блока

На данном графике видно, что производительность процессора увеличивается при увеличении размера блока, пока не наступает переполнение кэш-памяти. После этого происходит очевидное падение производительности.

С помощью программы LAPTEST были измерены производительности выполнения различных операций при использовании разного числа ядер (от двух ядер до максимального числа ядер одного узла и нескольких узлов). Замерялось время выполнения операций, и вычислялась средняя производительность одного ядра процессора. Данные позволяют сделать вывод об использовании кэш-памяти.

Производительность выполнения операции DGEMM (BLAS Level 3) с размером матрицы 256 ($C = 0.5 * A * B + C$) практически не зависела от числа используемых ядер в диапазоне от 2 до 12 ядер процессоров Intel Xeon 5675 (Westmere) на двух узлах и составляла около 11.7 GFlops. Однако, при использовании 24 ядер производительность снизилась до 10.5 GFlops, что объясняется совместным использованием кэш-памяти.

Рассмотрим производительность при выполнении операций чтения и записи в память при использовании буфера размера в 500 MB:

Операция *Memory 1* — векторная операция, связанная с чтением из памяти длинных векторов: x = x + B[1:25685333] * C[1:25685333]

Операция *Memory 2*:- операция записи в память длинного вектора A[1:25685333] = x.

Производительность данных операций зависела от числа используемых ядер процессоров Westmere (рис. 6) .

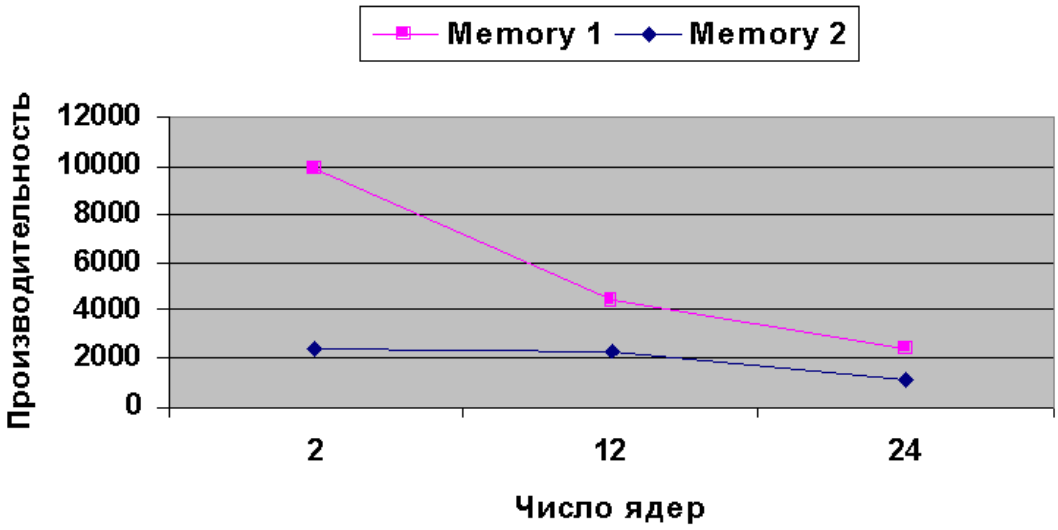

Рис. 6. Зависимость производительности (МБ/с) одного ядра от числа используемых ядер

Производительность при выполнении операций Memory 1и Memory 2: представлена на рис. 7 и 8 для последовательного обращения к элементам массива. Известно, что обращение к памяти в вычислительных системах оптимизировано под последовательный доступ. Поэтому проводилось тестирование операций с памятью с шагом 1031, результаты которого представлены на рис. 9 и 10. Графики отображают зависимость производительности от объема используемой памяти.

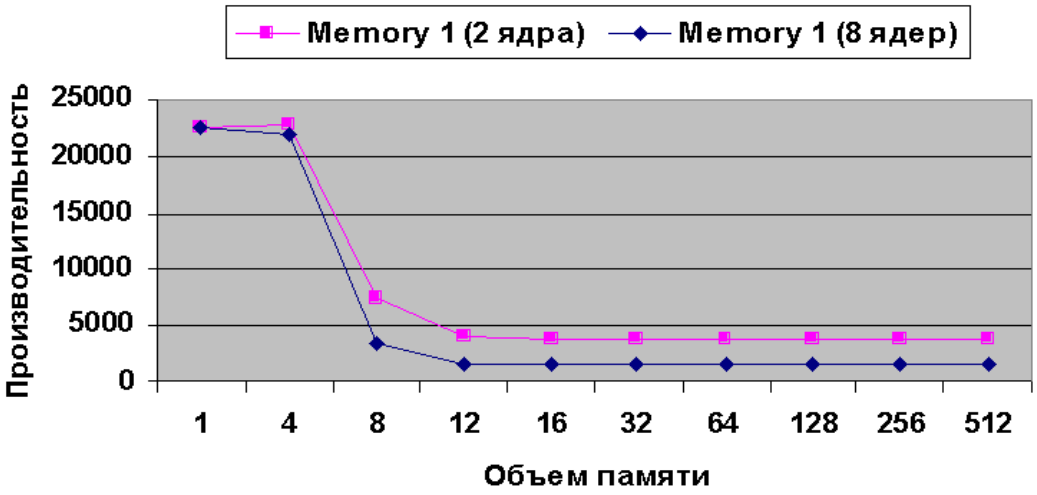

Рис. 7. Чтение из памяти. Зависимость производительности (МБ/с) одного ядра от объема используемой памяти (МБ)

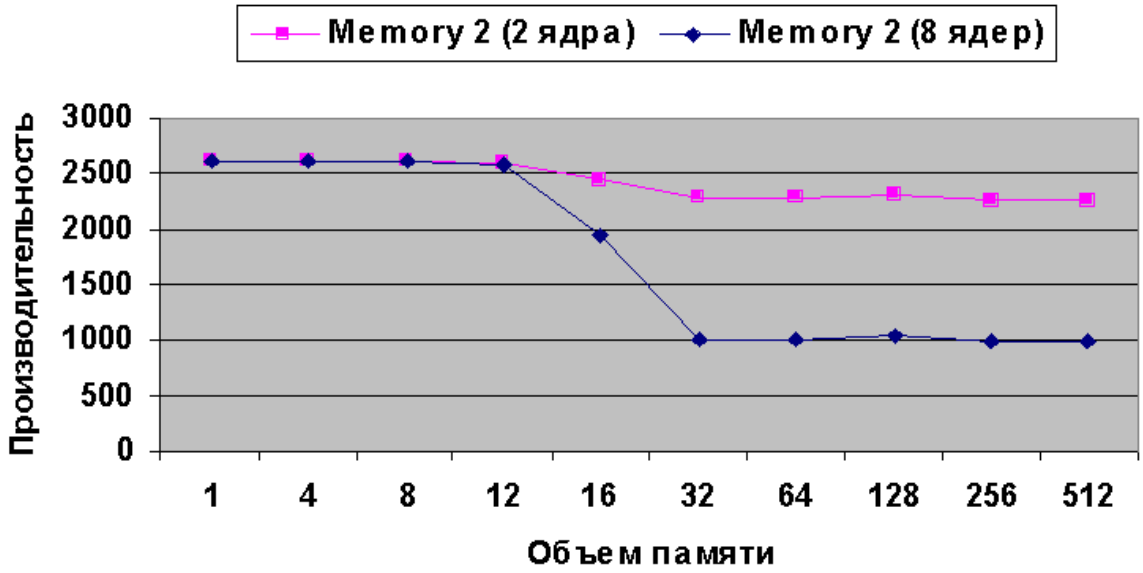

Рис. 8. Запись в память. Зависимость производительности (МБ/с) одного ядра от объема используемой памяти (МБ)

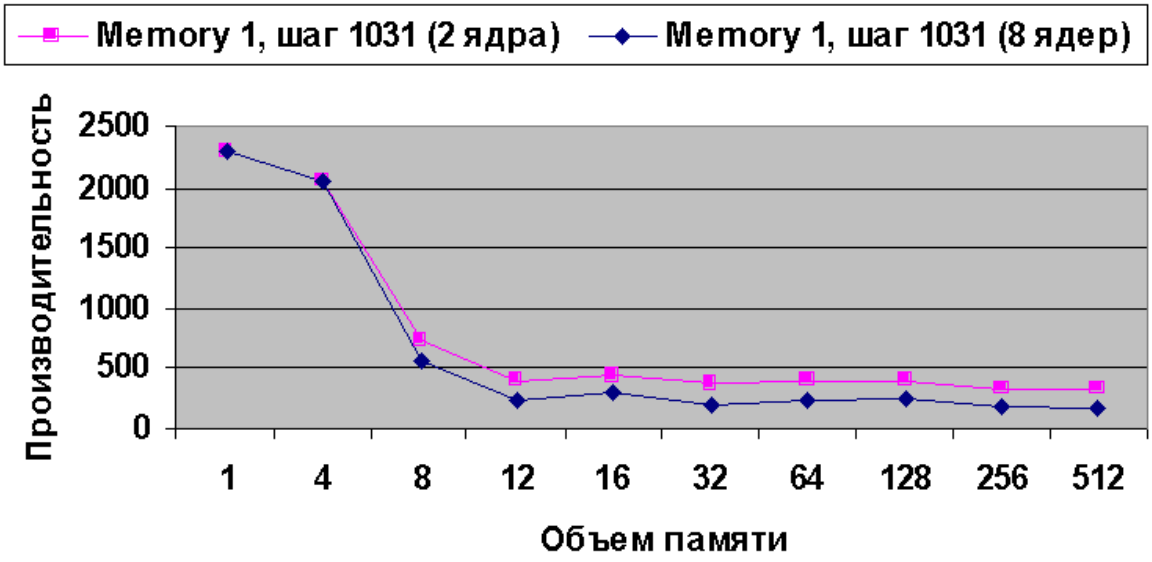

Рис. 9. Непоследовательное чтение из памяти. Зависимость производительности (МБ/с) одного ядра от объема используемой памяти (МБ)

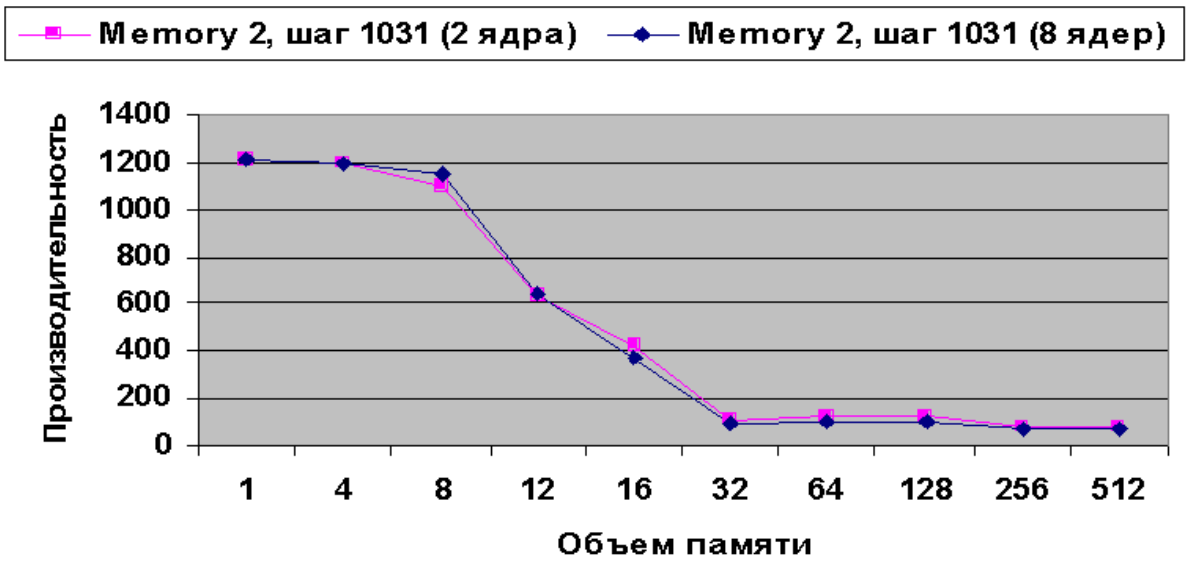

Рис. 10. Непоследовательная запись в память. Зависимость производительности (МБ/с) одного ядра от объема используемой памяти (МБ)

Исходя из полученных данных можно рассчитать определить замедление, вызванное использованием кэш-памяти. Для процессора Harpertown при длине вектора 2000000 оно составляет 9.7, для процессора Westmere при той же длине вектора оно составляет 3.46. При этом замедление, вызванное непоследовательным чтением из памяти, составляет десятичный порядок.

Исследование особенностей использования многоядерных процессоров является важной задачей для расчета оптимальных параметров параллельных программ и для построения априорных оценок эффективности различных вариантов распараллеливания в системах автоматического и автоматизированного распараллеливания. Наиболее сильный эффект проявляется при превышении размеров кэш-памяти и при конфликтах обращений к памяти, когда используется много ядер. Это следует учитывать при оптимизации программ.

Работа выполнена при поддержке госконтракта Министерства образования и науки РФ № 07.514.12.4001.

ЛИТЕРАТУРА:

1. Суперкомпьютер “МВС-100К”: сайт. URL: http://www.jscc.ru/hard/mvs100k.shtml (дата обращения 30.05.2012)
2. D. Eadline “Optimized HPC Performance: MPI Strategies for Next Generation Quad-Core Intel®Xeon® Processors”. URL: http://basement-supercomputing.com/download/reports/harpertown-WP1-rev3.pdf (дата обращения 30.05.2012)
3. Г.И. Савин, П.Н. Телегин, Б.М. Шабанов “Кластеры Беовульф” // Известия высших учебных заведений. Электроника. 2004, № 1, С. 7-12
4. Б.М. Шабанов, П.Н. Телегин, О.С. Аладышев “Особенности использования многоядерных процессоров” // Программные продукты и системы. 2008, № 1, С. 7-9
5. П.Н. Телегин, Б.М. Шабанов “Связь моделей программирования и архитектуры параллельных вычислительных систем” // Программные продукты и системы. 2007, № 2, С. 5-8
6. В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина “Автоматическое распараллеливание последовательных программ для многоядерных кластеров” // Труды Всероссийской научной конференции “Научный сервис в сети Интернет: суперкомпьютерные центры и задачи”, Новороссийск, сентябрь 2010, М.: Изд-во МГУ, 2010, С. 12-15
7. BLAS (Basic Linear Algebra Subprograms). URL: http://www.netlib.org/blas
8. С.Ю. Лапшина "Высокопроизводительные вычисления в практике моделирования роста перколяционных кластеров" // Программные продукты и системы. – 2011. - № 4 (96). – с.75-79.