3. *Жаров В. П., Латышев А. С.* Лазерно-ультразвуковые технологии в медицине // Лазерная медицина. 1999. Т.3, вып.1. С.4–11.
4. *Вишневский В. А.* Низкоинтенсивное лазерное излучение в лечении остеоартроза // Вестник морской медицины. 1999. №4(12). С.25–32.
5. *Плетнев С. Д.* Лазеры в клинической медицине. М.:Медицина, 1996. 428с.
6. *Кару Т. Й.* Изменение спектра поглощения монослоя живых клеток после низкоинтенсивного лазерного облучения // ДАН 1998. Т.360, №2. С.267–270.
7. *Зубкова С.М*. О механизме биологического действия излучения гелий-неонового лазера // Биологические науки. 1978. № 7. С.30–37.
8. Мембранотропное действие низкоэнергетического лазерного облучения крови / А. А. Спасов, В. В. Негода, О. В. Островский, К. Конан // Бюллетень экспер. биол. и мед. 1998. Т.126, № 10. С.412–415.
9. *Воронина О. Ю., Каплан М. А., Степанов В. А.* Воздействие низкоинтенсивного лазерного излучения на биоткани // Письма в ЖТФ. 1990. Т. 16, вып. 6. С.46–49.
10. *Железнякова Т. А*. Исследование влияния низкоинтенсивного лазерного излучения на биодоступность антибиотиков // Физика конденсированного состояния: Тез. докл. XII Республ. науч. конф. аспирантов, магистрантов и студентов. Гродно: ГрГУ, 2004. С.343-345.

РАЗРАБОТКА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КОММУНИКАЦИОННОГО ПРОЦЕССА В MPI

О. А. Зиневич

Параллелизм – это возможность одновременного выполнения нескольких арифметико-логических или служебных операций [1]. Его использование дает эффективный инструмент при решении задач, требующих огромных вычислительных затрат. Однако, для того чтобы параллельная программа давала ощутимый выигрыш в скорости в сравнении с последовательной проектировщикам приходится решать ряд специфических проблем. Прежде всего, это равномерная загрузка процессоров, т. к. если основная вычислительная работа ложиться только на часть из них, уменьшиться и выигрыш от распараллеливания. Другая не менее важная проблема – эффективная организация обмена информацией между процессами. Если вычисления выполняются на высокопроизводительных процессорах, загрузка которых достаточно равномерна, но скорость обмена данными при этом низка, большая часть времени будет тратиться впустую на ожидание информации, необходимой для дальнейшей работы. Это может существенно снизить скорость работы программы [2].

Наиболее достоверным способом проверки эффективности любой параллельной программы является тестирование ее на конкретной вычислительной системе. Но этот метод требует пройти все стадии разработки, что является довольно трудоемким и дорогостоящим процессом и может занять значительное время. Поэтому предлагается эту задачу упростить,

используя для предварительного анализа формульную модель. Она позволит отсеять алгоритмы очевидно не удовлетворяющие поставленным задачам и выбрать лучший из оставшихся.

Формульная модель строится на основе сетевого закона Амдала, который выглядит следующим образом [3]:

$$R = T_1 / T_n = \frac{W \cdot t}{\left(W_{\tilde{n}\hat{e}} + \frac{W_{\ddot{i}\,\eth}}{n} \right) \cdot t + W_c \cdot t_c} = \frac{1}{a + \frac{1-a}{n} + \frac{W_c \cdot t_c}{W \cdot t}} = \frac{1}{a + \frac{1-a}{n} + c}, \quad (1)$$

где R – ускорение параллельной системы, T_1 – время решения задачи на однопроцессорной системе, а T_n – время решения той же задачи на n-процессорной системе, $W=W_{ск}+W_{пр}$ – общее число операций в задаче, $W_{пр}$ – число операций, которые можно выполнять параллельно, а $W_{ск}$ – число скалярных (нераспараллеливаемых) операций, $a=W_{ск}/W$ – удельный вес скалярных операций, W_c – количество передач данных, t_c – время одной передачи данных, $c = (W_c \cdot t_c)/(W \cdot t)$ – коэффициент сетевой деградации вычислений. Если воспользоваться довольно несложными рассуждениями, то закон Амдала легко трансформируется в искомую формулу, которая зависит от известных параметров вычислительной системы.

Для определенности формульная модель была спроектирована для параллельного алгоритма решения СЛАУ методом Гаусса-Зейделя, при этом, поскольку для систем с большим объемом вычислений можно пренебречь потерями времени на вспомогательные операции полагалось, что a=0, т. е. $W_{ск}$=0 и $W=W_{пр}$. В действительности рассчитывалось время выполнения одной итерации на n-процессорной системе T_n, а ускорение находилось из соотношения T_1/T_n. Формула имеет следующий вид:

$$T_n = \frac{m \cdot (t_{дел} + (m-1)t_{умн} + (m-1)t_{сл})}{n} \cdot \frac{1}{f} + \frac{8N}{S} + L \cdot \{\log_2 n\}, \quad (2)$$

где m – размерность СЛАУ, $t_{дел}$, $t_{умн}$, $t_{сл}$ – время выполнения арифметических операции в тактах, f – тактовая частота процессоров кластера, N – количество пересылаемых элементов каждым из процессов после очередной итерации, S – пропускная способность, L – латентность.

Данная формула была проверена экспериментально, и оказалось, что она действительно пригодна для первичной оценки работы параллельных программ. Однако, были выявлены также существенные недостатки такой модели: невозможность учета рассинхронизации процессов во время вычислений, некоторая неуниверсальность (при изменении алгоритма меняется и формула). Формула также несколько видоизменяется и при различной структуре коммуникационной сети. Формула (2) написана

для коммутатора, а, например, для концентратора необходимо домножить на *n* второе и третье слагаемое.

На рис. 1 представлены результаты формульного моделирования для кластера, построенного на сети Fast Ethernet процессорах с тактовой частотой 1ГГц. Видно, что кривая ускорения достигает максимума в некоторой точке и затем даже начинает спадать. Такая зависимость во многом объясняется влиянием обменов между процессами, так как при их исключении зависимость носит линейный характер. С улучшением характеристик сети и узлов, достигается большее максимальное ускорение. Однако, форма кривых, ограничивающая ускорение, сохраняется. Таким образом, чтобы сделать ускорение как можно более линейным надо искать алгоритмы, в которых обмен почти отсутствует. Таковыми являются, например вероятностные методы Монте-Карло.

Как было сказано и показано ранее обмен сообщениями играет важную роль в параллельных вычисления, поэтому важно знать, как организованы коммуникации в используемой параллельной библиотеке (в данном случае MPI). Для этого на рис. 2 представлены результаты эксперимента, в котором сравнивался стандарт MPI с популярными протоколами TCP и UDP. В эксперименте измерялось время передачи между двумя компьютерами от объема пересылаемых данных. Причем данные передавались туда и обратно по несколько раз. Из графика видно, что коммуникации в MPI реализованы на протоколе TCP. Хотя UDP и быстрее, он не обладает требуемой надежностью. На рис. 3 отображены результаты

схожего эксперимента для MPI и TCP, но здесь обмены происходили на фоне загруженного процессора (на обоих компьютерах была запущена антивирусная программа). Эффективность TCP почти не изменилась, в

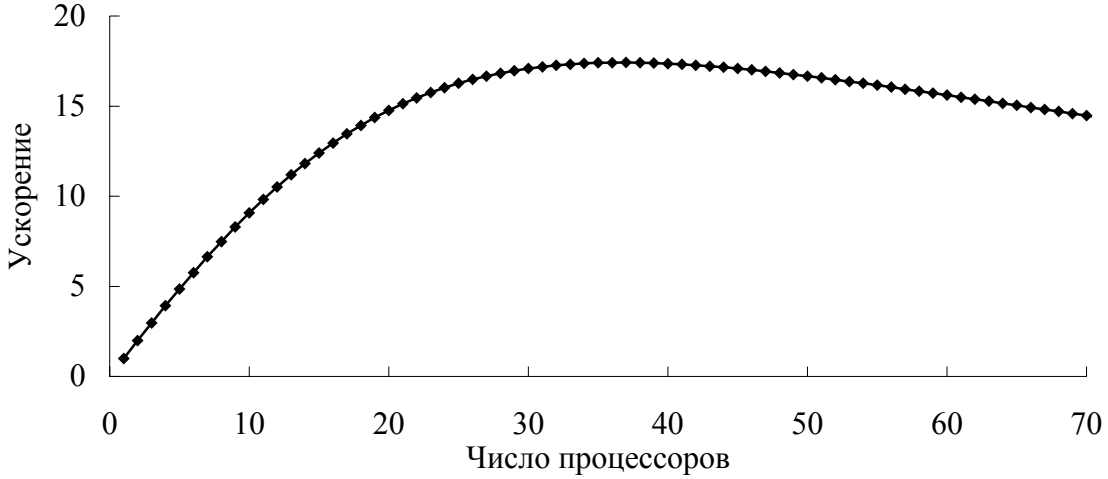

Рис. 1. Формульная зависимость ускорения от числа процессоров для Fast Ethernet

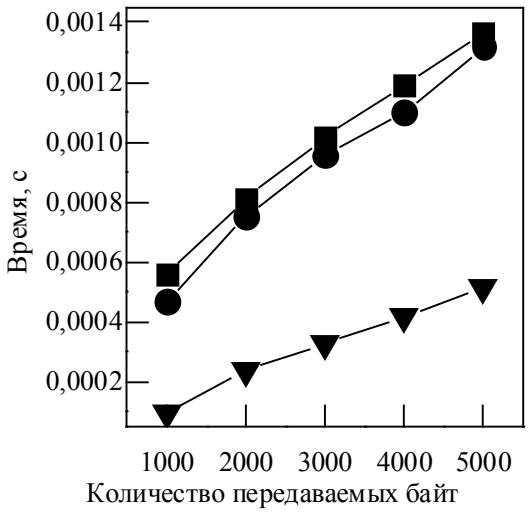

Рис. 2. Сравнение MPI, TCP и UDP: ■–MPI, ●–TCP, ▼–UDP

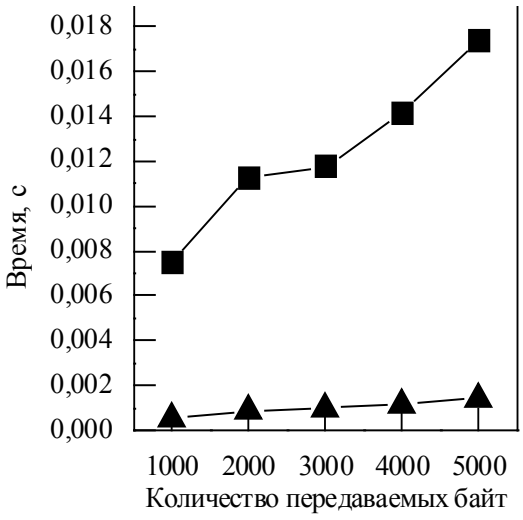

Рис. 3. Сравнение MPI и TCP при загрузке процессоров

то время как для MPI результаты заметно ухудшились. Можно сделать вывод, что коммуникационная библиотека MPI потребляет значительный ресурс процессора и TCP оказывается лучше порядком на 90%. В этом контексте MPI далек от совершенства.

В работе получена формульная модель для оценки эффективности параллельной программы. С ее помощью показано, что масштабируемость системы кардинальным образом зависит от объема пересылаемых данных и скорости передачи данных. Также из сравнения протоколов TCP и UDP с MPI сделан вывод, что в случае совмещения обменов с вычислениями MPI оказывается узким местом.

Литература

1. *Шпаковский Г. И., Серикова Н. В.* Программирование для многопроцессорных систем в стандарте MPI. – Мн.: БГУ, 2002. – 323 с.
2. *Немнюгин С. А., Стесик О. Л.* Параллельное программирование для многопроцессорных систем. – СПб.: БХВ-Петербург, 2002. – 400 с.
3. *Шпаковский Г. И.* Организация параллельных ЭВМ и суперскалярных процессоров: Учеб. пособие. Мн.: Белгосуниверситет, 1996. 284 с.

ДЕТЕКТИРОВАНИЕ ДВИЖЕНИЯ В СИСТЕМАХ ВИДЕОНАБЛЮДЕНИЯ

А. С. Кочетков

Для систем видеонаблюдения типичным является применение алгоритмов детектирования движения путем анализа разности между яркостями пикселей последовательно идущих кадров [1]. Существенным недостатком подобных методов является неустойчивость по отношению к