УДК 004.2

В.И. ШИНКАРЕНКО

ЗАВИСИМОСТЬ ВРЕМЕННОЙ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ И ПРОГРАММ ОБРАБОТКИ БОЛЬШИХ ОБЪЕМОВ ДАННЫХ ОТ ИХ КЭШИРОВАНИЯ

Abstract: *Computer family that is forecasting linearly relative to running time of the programs; degree of the caching of data are a new concepts which introduced. It was shown that the use of probabilistic estimation of algorithms time's difficulty for choice of the algorithm from alternative at criterion of time's efficiency is insufficiently account for experimental data. It's reported that computer family with processor Intel and similar is the family for which forecasting of running time of the programs is essentially non-linear. Method of the qualitative estimation of the influence degree of caching data to the time's efficiency of algorithms was offered.*
Key words: *algorithm, time's efficiency, time's difficulty=calculate difficulty, caching data, degree of the caching.*

Анотація: *Вводяться поняття: сімейства ЕОМ лінійно-прогнозованих за часом виконання програм; ступінь кешування даних. Спираючись на експериментальні дані, показано недостатність імовірнісних оцінок часової складності алгоритмів для вибору алгоритму серед альтернативних за критерієм часової ефективності. Показана суттєва нелінійність прогнозування за часом виконання програм ЕОМ з процесорами Intel та споріднених. Запропонована методика якісної оцінки впливу ступеня кешування даних на часову ефективність алгоритмів.*
Ключові слова: *алгоритм, часова складність=обчислювальна складність, часова ефективність, кешування, ступінь кешування.*

Аннотация: *Вводятся понятия: семейства ЭВМ линейно-прогнозируемых по времени выполнения программ; степень кэширования данных. На основе экспериментальных данных показана недостаточность вероятностных оценок временной сложности алгоритмов для выбора алгоритма из числа альтернативных по критерию временной эффективности. Показана существенная нелинейность прогнозирования по времени выполнения программ ЭВМ на базе процессоров Intel и аналогов. Предложена методика качественной оценки влияния степени кэширования данных на временную эффективность алгоритмов.*
Ключевые слова: *алгоритм, временная сложность=вычислительная сложность, временная эффективность, кэширование, степень кэширования.*

1. Введение

Одной из существенных возможностей повышения эффективности программного обеспечения является подбор либо разработка эффективных алгоритмов. Возникает задача оценки их эффективности и, в первую очередь, временной эффективности алгоритмов [1].

Классическим является вероятностный подход к оценке временной эффективности алгоритмов. Общепринятой характеристикой алгоритма является вычислительная сложность [2], которая и используется для прогноза временной эффективности. Вычислительная (или временная) сложность алгоритма оценивается либо функцией зависимости количества операций от объема данных, либо функцией, ограничивающей ее сверху (асимптотическая временная сложность) [2–12 и др.].

Такой подход является продуктивным при разработке алгоритмов. Для применения же алгоритмов в конкретном программном обеспечении, предусматривающих их функционирование в конкретных условиях, такие оценки не всегда достаточны. Это, в частности, связано с особенностями исполнительных механизмов, выполняющих алгоритмы.

Назовем семейство ЭВМ линейно-прогнозируемым по времени выполнения программ (ЛПВ – семейство), если для любых двух ЭВМ из этого семейства существуют такие a и b, что для любой вычислительной программы справедлива линейная зависимость:

$$t_1 = at_2 + b, \qquad (1)$$

где t_1, t_2 – время выполнения программы на одной и другой ЭВМ соответственно в режиме предоставления всех ресурсов ЭВМ в их исключительное использование. Под вычислительной программой подразумевается однопотоковая программа в части, не использующей обмен данными посредством технических устройств ввода/вывода, без задержек и системных вызовов. При условии выполнения соотношения (1) в семейство могут быть включены ЭВМ различной архитектуры. ЭВМ, которые не входят в ЛПВ, семейство образуют НЛПВ – семейство.

Среди массово используемых ЭВМ ЛПВ-семейство образует ЭВМ серии ЕС, ПЭВМ с процессорами 8086 и другие, в основном, морально устаревшие модели.

Современные ЭВМ, в основном, являются *существенно* не линейно-прогнозируемыми по времени выполнения программ [13, 14]. Можно говорить о существенной непредсказуемости времени выполнения алгоритмов и программы применительно к исполнительному механизму.

Очевидно, что вероятностные оценки временной эффективности, накладываясь на нелинейность прогноза времени выполнения программ на конкретной модели ЭВМ, теряют в какой-то степени информативность. Появляется некая нелинейная зависимость между вычислительной сложностью и временной эффективностью, ввиду чего нельзя подменять показатель качества алгоритмов "временная эффективность" показателем "вычислительная сложность".

Причины временной непредсказуемости ЭВМ (или сложности прогноза) заключаются в конструктивных особенностях архитектуры процессоров и памяти. К таким особенностям относятся конвейерная обработка команд, многоуровневая организация памяти и, в общем, сложность всех технических средств с их многочисленными модификациями.

Конструктивные решения, заложенные в конвейере, а именно: количество конвейеров, распределение потока команд между ними, латентность конвейера, предконвейерная оптимизация, прогноз переходов и др. непредсказуемо (с точки зрения количественной оценки) сказываются на производительности процессора. К тому же сама производительность существенно зависит от входного потока команд и данных. В конечном итоге это влияет на временную эффективность программ при их выполнении на компьютерах таких архитектур.

На время выполнения программ, безусловно, оказывают влияние объем и порядок использования регистровой, кэш-памяти, оперативной и виртуальной (на внешних носителях) памяти.

Остановимся на одной из подсистем современной ЭВМ, одной из составляющих многоуровневой памяти – кэш-памяти.

Учитывая то, что именно алгоритм, а не программа и ее реализация, предопределяет последовательность обработки данных, закономерно возникает задача качественной и количественной оценки влияния порядка использования данных на временную эффективность алгоритмов с учетом размещения данных в ОП и/или кэш-памяти. Тем более, что управление размещением данных в многоуровневой памяти выполняется техническими средствами, без вмешательства разработчиков программ.

Иначе говоря, размещение данных в кэш-памяти с программной стороны предопределяется алгоритмом и структурами данных, а с технической – аппаратными средствами управления кэш-памятью. Какая эффективность такой программно-аппаратной системы?

Особенности временной эффективности обработки структур данных с учетом кэширования в некоторой степени рассмотрены в [13]. Данная работа направлена на исследование временной эффективности алгоритмов.

2. Предварительный анализ

Эффективность применения кэш-памяти не вызывает сомнений. Достаточно отключить в BIOS кэш L2, как снижение производительности процессора станет ощутимым, отключение же кэш-памяти L1 затормозит его работу настолько, что на лучших компьютерах на базе процессоров Intel практически работать будет невозможно. Вопрос в другом: насколько кэширование данных может повысить временную эффективность алгоритмов.

Почему временная эффективность алгоритмов зависит от кэширования данных?

Алгоритм предполагает некоторую последовательность операций с данными. Данные же могут находиться на разных уровнях многоуровневой памяти. Время доступа к данным на разных уровнях существенно различно. Так, время доступа к данным в кэш-памяти может быть на два порядка выше, чем к данным в ОП [13, 14], а к данным в ОП – на 3-4 порядка выше, чем к данным виртуальной памяти на магнитном диске.

Естественно, что непременным условием эффективной работы алгоритмов рассматриваемого класса (алгоритмов обработки больших объемов данных) является требование к размещению всех их данных в ОП, и в большинстве случаев технические средства это позволяют. С другой стороны, также в большинстве случаев, они не могут быть полностью помещены в кэш-память, что обусловлено гораздо меньшими размерами кэша и совместное использование ее всеми задачами в многозадачных операционных системах.

Как известно [14], при обращении к данным в ОП, они загружаются в кэш. При этом загружаются не только затребованные данные, но и соседние – загружается линейка кэша (32–64 байта). Повторное обращение как к затребованным данным, так и к "попутно" загруженным в кэш, значительно сокращает время выполнения программы за счет сокращения времени доступа. Следует отметить, что ввиду ограниченности кэша, загрузка новой линейки кэша практически всегда сопровождается вытеснением некоторой другой. Какой именно, зависит от алгоритма управления кэшем, который может учитывать частоту обращения к данным линейки, время ее нахождения в кэше, требования совместного использования кэша несколькими процессами (за ними, например, может "бронироваться" некоторое место). При этом алгоритм обработки данных как бы "сотрудничает" с алгоритмом управления кэшем. Насколько эффективно это сотрудничество?

Допустим, некоторый алгоритм последовательно сканирует данные, последовательно расположенные в ОП. При этом частота попадания в кэш будет максимальной потому, что всегда все "попутно" загруженные данные будут в кэше (не будут вытеснены другими). Алгоритм будет нести минимальные временные потери, связанные с доступом к данным.

Если другой алгоритм, функционально эквивалентный первому, будет обращаться к тем же данным, но вразброс, в произвольном порядке количество попаданий уменьшится и соответственно увеличатся временные затраты на доступ к данным. Хотя второй алгоритм может

оказаться и эффективнее за счет снижения количества операций. С другой стороны, если общий объем данных будет меньше размера кэша и они не вытесняются по требованиям других процессов, потерь на повторную загрузку данных в кэш может и не быть. Следовательно, увеличение временных потерь на доступ к данным следует ожидать, когда объем обрабатываемых данных превысит размер кэша (или его доступную часть).

Будем говорить об алгоритмах, а не программах в силу того, что именно алгоритм предопределяет последовательность обработки данных. Программа конкретизирует алгоритм относительно структур данных. А именно последовательность обработки является наиболее существенным в рассматриваемом контексте. С другой стороны, в [15] определяется: программа – есть алгоритм.

3. Прямой численный эксперимент

В случае, когда объем данных превышает размер кэш-памяти, временная эффективность алгоритмов снижается ввиду растущего числа промахов в кэш.

Прямой численный эксперимент выполнен с целью экспериментального подтверждения самого факта соответствующего изменения временной эффективности и качественной оценки степени такого изменения.

Строятся экспериментальные зависимости среднего времени выполнения алгоритма от объема обрабатываемых данных. Объем данных изменяется с некоторым шагом в интервале с центром, соответствующим размеру кэша.

Исследование выполнялось на алгоритмах сортировки. Они достаточно исследованы и являются репрезентативными представителями алгоритмов обработки больших объемов данных. Во избежание разночтений и подробных спецификаций самих алгоритмов выбраны алгоритмы сортировки, приведенные Н. Виртом в [2] с идентичными текстами программ на ПАСКАЛе:

A_1 – быстрая сортировка [2, стр. 99];

A_2 – пирамидальная сортировка [2, стр. 95];

A_3 – сортировка простым включением[2, стр. 79];

A_4 – Шейкер-сортировка [2, стр. 86];

A_5 – сортировка простым выбором [2, стр. 82];

A_6 – сортировка пузырьком [2, стр. 84], минимально модифицированная с отслеживанием момента завершения;

A_7 – сортировка бинарным включением [2, стр. 80];

A_8 – сортировка Шелла [2, стр. 99] с рекомендованной Д. Кнутом последовательностью приращений 1,3,7,15,31.

Экспериментальная база с технической стороны состояла из компьютеров различной конфигурации, различающихся объемом кэш-памяти, частотой процессора и шины и другими техническими характеристиками. Некоторые из них приведены далее в таком порядке – процессор /

чипсет системной платы – кэш L1 кода / L1 данных / L2, Кб – тактовая частота/частота системной шины / частота памяти, МГц – время доступа к ОП: чтение/запись, Мб/с:

1. Intel Pentium IIIE / Intel Solano i815E - 16/16/256 – 650 (6.5 x 100) / 100 / 133 - 515/148.

2. Intel Celeron-S / Intel Solano i815E - 16/16/256 - 994 MHz (10 x 99) / 99 /133 - 507/120.

3. Intel Pentium 4 /Intel Brookdale i845D - 12/8/256 - 1600 MHz (4 x 400)/133/133 - 1449/583.

4. Intel Pentium 4HT/ Intel Springdale-G i865G - 12/8/512 - 2400 MHz (3 x 800) / 200/200 - 3794/1205.

5. Intel Pentium 4A / Intel Brookdale i845E - 12/8/512 - 2400 MHz (4.5 x 533)/133/166 - 1911/667.

6. Intel Pentium 4E / Intel Springdale-G i865G - 12/16/1024 - 2400 MHz (4.5 x 533)/166/200 - 3298/1186.

7. AMD Duron XP / VIA VT8375 ProSavage DDR KM266 - 64/64/64 - 1400 (5.25 x 267)/133/200 - 1215/438.

8. AMD Duron XP / nVIDIA nForce2 Ultra 400 - 64/64/256 - 1666MHz (5 x 333)/166/200 - 2205/815.

Эксперименты выполнялись в MS DOS 6.22, что исключило влияние многозадачности операционных систем.

Программа транслировалась в среде Delphi 7.0. Для выполнения в DOS использовалась программа WDOSX (M. Типач), конвертирующая EXE модуль из формата PE (Windows) в формат MZ (DOS) с заменой соответствующих функций из системных библиотек. Таким образом решалась проблема ограничения сегмента данных и 64К (а, соответственно, и максимального объема обрабатываемых данных) в реальном режиме процессора Intel и отсутствия при этом компилятора с ПАСКАЛя в DOS под защищенный режим [16], в котором такого ограничения нет.

Время конкретного выполнения программы замерялось в тактах процессора путем снятия значений регистра счетчика времени. Перед каждым выполнением алгоритмов кэш-память очищалась интенсивной работой с большим посторонним массивом данных и процессорной командой очистки кэша WBINVD.

Таким образом, было получено более 100 зависимостей времени сортировки от объема данных, анализ которых позволяет сделать некоторые обобщения.

Зависимости времени сортировки целых 4-байтовых чисел (integer) на ПЭВМ №1, приведенные на рис. 1, являются достаточно типичными для других ПЭВМ. Для наглядности временные значения нормированы.

Можно заметить, что скорость изменения времени выполнения алгоритмов (угол наклона кривых) существенно изменяется при объеме данных (L_d), превышающих размер кэша (L_k) для алгоритмов $A_3 ... A_8$ и почти не изменяется для алгоритмов A_1, A_2. Очевидно, такое изменение связано с изменением степени кэширования данных.

Назовем степенью кэширования данных при выполнении некоторого алгоритма и заданных структурах данных следующее:

$$PK = \frac{n_+ - n_-}{n_+ + n_-}, \qquad (2)$$

где n_+, n_- – количество попаданий и промахов в кэш соответственно.

При $L_d < L_k$ PK близко к единице, а если L_d начинает превышать L_k, PK постепенно снижается до -1.

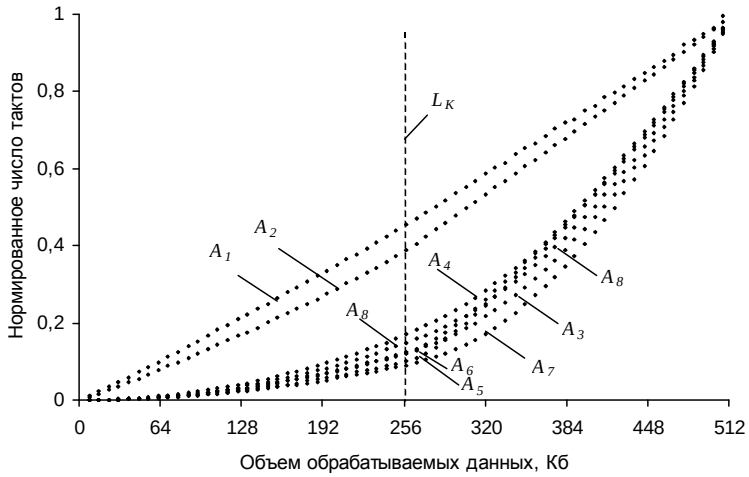

Рис. 1. Зависимость времени сортировки алгоритмов $A_1 ... A_8$ от размера сортируемого массива

Оценивая качество изменения функций (на рис. 1 и аналогичных), следует отметить, что алгоритмы в разной степени "чувствительны" к изменению PK. Хотя для принятия решений необходимы количественные оценки, в данной работе влияние кэширования данных на временную эффективность алгоритмов исследовано с качественной стороны.

Отметим, что на время выполнения алгоритма с технической стороны влияет не только кэширование данных, но и множество других технических характеристик: глубина конвейера, соотношение частоты шины и частоты процессора, тип оперативной памяти и многие другие. В подтверждение этого приведем характерную зависимость времени выполнения одного из алгоритмов (A_3) от L_d на компьютерах экспериментальной базы с одинаковым размером кэша L2 (рис. 2). Влияние степени кэширования на различных ЭВМ оказывается различным. Время выполнения алгоритма A_3 при объеме данных 512 Кб варьируется в пределах $3 \cdot 10^{10} ... 6 \cdot 10^{10}$.

Отметим также, что количество тактов, необходимых для выполнения одного и того же алгоритма и при тех же структурах данных, отличается более чем в два раза на различных компьютерах (а при больших объемах этот разрыв может составлять десятки и сотни раз). Это еще раз подтверждает необходимость учета архитектуры и модификаций ЭВМ при подборе технических средств, для выполнения алгоритмов, обрабатывающих большие объемы данных и критичных по времени.

Еще один аргумент в пользу учета архитектуры ЭВМ при выборе алгоритма приведен на рис. 3, где видно, что при L_d =450 … 512К на ЭВМ № 1 быстрее выполняется алгоритм A_4, а на ЭВМ № 2 - A_5. Приведенные данные измерялись тактами процессора. В процессе исследований получено достаточно много подобных примеров как в измерении тактами, так и в реальном времени.

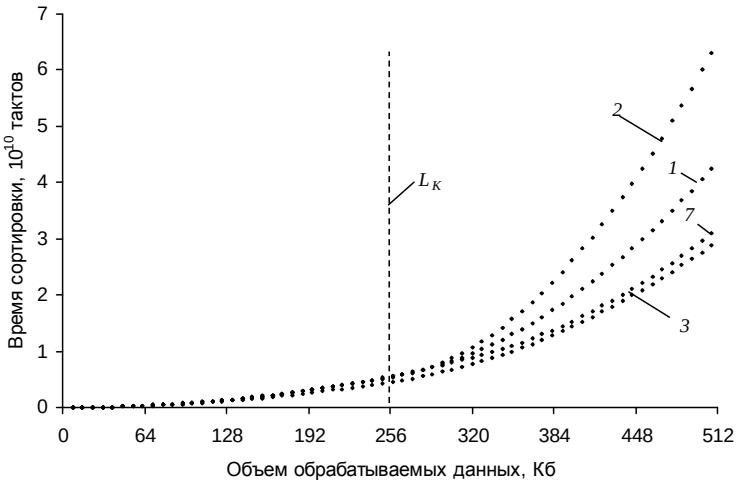

Рис. 2. Зависимость времени сортировки A_3 от размера сортируемого массива на различных ЭВМ (сносками указаны номера компьютеров экспериментальной базы)

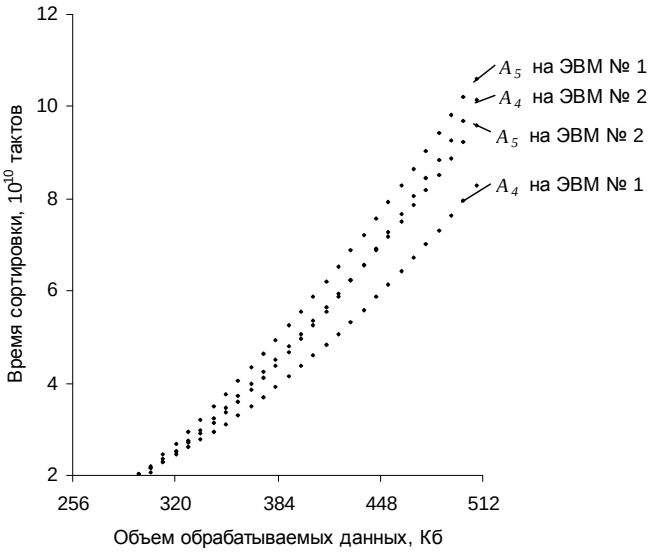

Рис. 3. Зависимость времени сортировки A_4 и A_5 от размера сортируемого массива

Заметим также, что хотя вероятностные зависимости [2, 3] для алгоритмов $A_3 ... A_7$ являются зависимостями второго порядка, экспериментальные зависимости существенно отличаются от квадратичной.

4. Сопоставление вероятностных оценок и результатов прямых экспериментов

Рассмотрим оценку вычислительной сложности алгоритма сортировки простыми вставками (A_3). Среднее количество сравнений (N_{cp}) и присваиваний (пересылок N_{np}) определяется как [2]

$$N_{cp} = \frac{1}{4}(n^2 + n - 2);$$
$$N_{np} = \frac{1}{4}(n^2 + 9n - 10), \qquad (3)$$

где n – количество элементов для сортируемых массивов данных.

Если известно, что время операции сравнения $t_{ср}$ и присваивания $t_{пр}$, то можно считать, что время выполнения алгоритма (для ЭВМ ЛПВ семейства)

$$t_{A_3} = \frac{1}{4}(n^2 t_{ср}^2 + nt_{ср} - 2 + n^2 t_{пр}^2 + 9nt_{пр} - 10). \quad (4)$$

Так как время выполнения операций сравнения и присваивания непостоянно (из-за возможного разного времени доступа к данным, конвейерной обработки команд и т.д.), можно говорить лишь о среднем времени $\bar{t}_{ср}$ и $\bar{t}_{пр}$. При этом, $\bar{t}_{ср}$ будет включать время на непосредственно сравнения и время на операции управления последовательностью действий (особенностью данного алгоритма является то, что количество сравнений совпадает с количеством выполнения внутреннего цикла). Внешний цикл выполняется n раз, и время на выполнение операций управления при количестве итераций более 2000 можно проигнорировать (при этом погрешность будет порядка 0,01 … 0,1%).

Используя полученные результаты экспериментов, определены $\bar{t}_{ср}$ и $\bar{t}_{пр}$ из условия минимального среднеквадратичного отклонения:

$$\varphi = \sum_i (t_i^T - t_i^Э)^2 \to \min, \quad (5)$$

где $t_i^T, t_i^Э$ – время сортировки i-массива, вычисленное по (4) и замеренное экспериментально.

Приравнивая к нулю частные производные согласно методу наименьших квадратов

$$\begin{cases} \dfrac{\partial \varphi}{\partial t_{ср}} = 0; \\ \dfrac{\partial \varphi}{\partial t_{пр}} = 0, \end{cases} \quad (6)$$

получаем систему нелинейных алгебраических уравнений 3-го порядка, аналитическое решение которой достаточно сложно. Поэтому значения $\bar{t}_{ср}$ и $\bar{t}_{пр}$ определялись методом случайного поиска, минимизировав φ.

По проведенным численным экспериментам на всех компьютерах экспериментальной базы получены вероятностные зависимости вида (4) с минимальным среднеквадратичным отклонением от экспериментальных данных. Пример такой зависимости для ЭВМ № 7 на рис. 4. Очевидно, что теоретическая кривая 2 не отражает характерных особенностей экспериментальной.

Качество аппроксимации будем оценивать по трем показателям: среднему и максимальному относительному отклонению теоретической кривой от экспериментальной; среднеквадратичному отклонению адекватности:

$$\sigma_a = \sqrt{\frac{1}{n-1}(\sum_{i=1}^{n} t_i^Э - t_i^T)^2}. \quad (7)$$

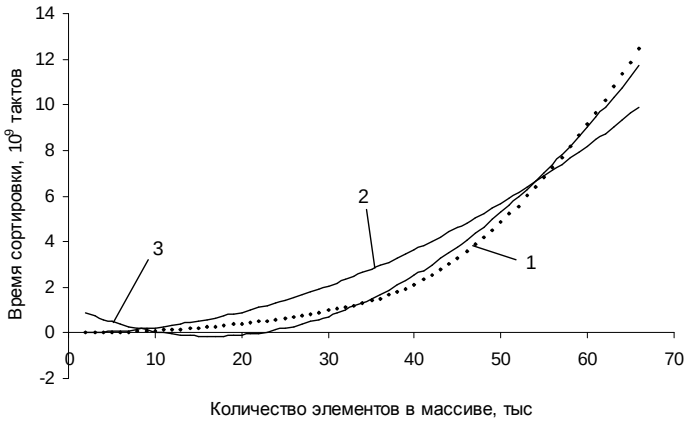

Рис. 4. Зависимость времени сортировки A_3 от размера массива. 1- экспериментальные данные, 2- теоретическая зависимость (4) со статистически определенными $\bar{t}_{cp}$ и $\bar{t}_{np}$, 3- парабола с минимальным среднеквадратичным отклонением от экспериментальных данных

Среднее относительное отклонение теоретической кривой от экспериментальной составило 78%, а максимальное – 128%, при среднеквадратичном отклонении равном $1{,}03*10^9$ (для замеров времени в пределах $0{,}005*10^9$ … $12{,}5*10^9$).

Теоретические зависимости алгоритма A_3 для остальных ЭВМ экспериментальной базы в 4 случаях из 8 близки к приведенным на рис.4 и - в лучшем случае дают отклонения 5%, 7% при значениях $0{,}006*10^9$ … $425*10^9$ – среднеквадратичное отклонение $5{,}97*10^9$, а в худшем – 90%, 134% при значениях $0{,}005*10^9$ … $66*10^9$ – среднеквадратичное отклонение $6{,}08*10^9$.

Точность теоретических зависимостей для алгоритмов $A_5 ... A_7$ мало отличается от приведенных результатов для A_3.

Исключением являются алгоритмы A_1 и A_2. Средняя относительная погрешность вероятностных зависимостей для A_1 не превышает 4% (в основном, не более 1%), для A_2 - 25% (в основном, порядка 10%). Среднеквадратичное отклонение порядка $1*10^5$ … $3*10^5$ при измеряемых значениях $3*10^5$ … $2*10^8$. Точность аппроксимирующей кривой намного выше. Это объясняется тем, что время выполнения примитивных операций для алгоритма A_1, вычисленное согласно (5) на компьютерах экспериментальной базы, составило 11 ..21 такт процессора, A_2 - 27 … 43 такта, в то время как для остальных алгоритмов – 0 … 3,3 такта. Столь большое время примитивных операций алгоритмов A_1 и A_2 можно объяснить лишь тем, что это время включает в себя и значительное время на доступ к данным. Увеличение времени доступа к данным в связи с увеличением количества промахов в кэш относительно мало сказывается на среднем времени доступа к данным. Поэтому ощутимого изменения характера зависимости $t(n)$ при $L_d > L_K$ не наблюдается.

Ввиду недостаточной точности предыдущей аппроксимации попробуем решить задачу аппроксимации экспериментальных данных другим способом.

Преобразуем (4):

$$t_{A_3} = \frac{1}{4}(\bar{t}_{cp}^2 + \bar{t}_{np}^2)n^2 + \frac{1}{4}(\bar{t}_{cp} + 9t_{np})n - \frac{1}{4}(2\bar{t}_{cp} + 10\bar{t}_{np}) = an^2 + bn + c\,, \quad (8)$$

где $a = \frac{1}{4}(\bar{t}_{cp}^2 + \bar{t}_{np}^2)$ $b = \frac{1}{4}(\bar{t}_{cp} + 9t_{np})$ $c = -\frac{1}{4}(2\bar{t}_{cp} + 10\bar{t}_{np})$. (9)

Определим коэффициенты a, b и c методом наименьших квадратов.

Построенные таким образом зависимости естественно лучше по критерию среднеквадратичного отклонения (в несколько раз) относительно первого способа. Однако максимальная относительная погрешность в этом случае значительно больше (часто на 1…2 порядка). Пример: кривая 3 на рис. 4. Заметно значительное расхождение кривых по форме.

Такой точности оценок времени выполнения алгоритмов, построенных на основании вероятностных оценок вычислительной сложности, зачастую недостаточно для принятия решений, связанных с выбором алгоритмов, так как соотношения времени выполнения алгоритмов могут быть значительно меньше имеющихся погрешностей (примеры на рис. 2, 3). Тем более, что при этом $\bar{t}_{cp}$ и $\bar{t}_{np}$ определялись статистически. Для ЭВМ, НЛПВ – семейства вероятностными методами определить время выполнения программ невозможно.

5. Уточнение зависимостей

Как уже отмечалось, при объеме данных $L_d < L_K$ при условии, что в кэш хранятся данные исключительно только исследуемого алгоритма, *PK*=1 и рост времени выполнения алгоритма, в основном, обусловлен изменением объема данных. При $L_d > L_K$ к этому фактору добавляется изменение *PK*, которое сопутствует изменению объема данных. Можно ожидать, что зависимости $t_{A_i}(n)$ до L_K и после L_K различны. Основываясь на этом, зависимость времени выполнения алгоритмов от количества элементов массива была найдена в виде кусочно-гладкой функции (2) или (3):

$$t(n) = \begin{cases} a_1 n^2 + b_1 n + c_1 \text{ при } n <= n_0 \\ a_2 n^2 + b_2 n + c_2 \text{ при } n > n_0 \end{cases}; \quad (10)$$

$$t(n) = \begin{cases} a_1 n\ln(n) + b_1 n + c_1 \text{ при } n <= n_0 \\ a_2 n\ln(n) + b_2 n + c_2 \text{ при } n > n_0 \end{cases}, \quad (11)$$

где n_0 – количество элементов массива, соответствующее точке изменения зависимости.

Выбор вида зависимости обусловлен вероятностными оценками временной сложности.

Для каждого алгоритма $A_1 ... A_8$ оценки параметров $a_1, b_1, c_1, a_2, b_2, c_2$ выполнялись методом наименьших квадратов. Вид теоретической зависимости (10) или (11) выбирался из условия минимума среднеквадратичного отклонения адекватности (7).

Точка n_0 определялась следующим образом (на примере зависимости вида (10)). Сначала определялась m:

$$m : \min_m \left(\sum_{i=1}^{m} (a_1 n_i^2 + b_1 n_i + c_1 - t_A^{\ni})^2 + \sum_{i=m}^{M} (a_2 n_i^2 + b_2 n_i + c_2 - t_A^{\ni})^2\right), \quad (12)$$

где M – количество массивов различных объемов данных в эксперименте. При этом для каждого m методом наименьших квадратов определялись коэффициенты $a_1, b_1, c_1, a_2, b_2, c_2$ из условия минимума функции:

$$\varphi = \sum_{i=1}^{m} (a_1 n_i^2 + b_1 n_i + c_1 - t_A^{\ni})^2 + \sum_{i=m}^{M} (a_2 n_i^2 + b_2 n_i + c_2 - t_A^{\ni})^2 \to \min.$$

Затем определялась n_0 как абсцисса точки пересечения двух кривых:

$$t_1(n) = a_1 n^2 + b_1 n + c_1 \text{ и } t_2(n) = a_2 n^2 + b_2 n + c_2. \quad (13)$$

Экспериментальные значения времени выполнения алгоритмов получены с соблюдением следующих условий:

измерения выполняются в среде, исключающей влияние посторонних процессов. В нашем случае в MS DOS 6.22;

перед предстартовым замером времени кэш очищался интенсивной работой с большим временным массивом и командой очистки кэша;

количество элементов в массиве изменялось с учетом объема кэша от нуля до $2L_K$ с шагом 1, 2, 4 или 8 тыс. элементов, в зависимости от L_K, при этом количество точек (размеров массива) было фиксировано (M =66);

время сортировки определялось как среднее из $N_{//}$ =9 параллельных испытаний, при каждом из которых массив повторно инициировался случайными значениями;

элементами массивов были целые 4-байтовые числа со знаком.

Полученные таким образом зависимости отличаются достаточно высокой точностью. Из 64 зависимостей (8 алгоритмов * 8 ЭВМ) среднее относительное отклонение находится в пределах 0,1 … 14,1%, что вполне допустимо для принятия решений в технической сфере.

Можно отметить, что практически во всех 64 случаях среднее относительное отклонение, максимальное относительное отклонение и среднеквадратичное отклонение не превосходят соответствующих значений для аппроксимаций, рассмотренных в п. 3.

Для сравнения, в табл. 1 приведены среднеквадратичные отклонения аппроксимирующих кривых при трех подходах к аппроксимации:

согласно вероятностным зависимостям [2, 3] со статистическим определением среднего времени выполнения базовых операций $\bar{t}_{ср}$ и $\bar{t}_{пр}$;

оптимальная по критерию минимального среднеквадратичного отклонения полиномиальная зависимость второго порядка;

оптимальная по критерию минимального среднеквадратичного отклонения зависимость вида (10), (11).

В табл. 1 приведены среднеквадратичные отклонения адекватности (7), средние по всем компьютерам экспериментальной базы.

На рис. 5 приведена типичная теоретическая зависимость (для алгоритма A_3 на ЭВМ № 2). Можно отметить хорошее совпадение теоретических и экспериментальных данных.

Таблица 1. Среднеквадратичные отклонения адекватности

	Среднеквадратичное отклонение, 10^6 тактов							
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
Аппроксимация 1	0,27	7,37	4092	-	3554	6016	4805	-
Аппроксимация 2	0,21	0,56	1839	1201	1171	2063	2208	1631
Аппроксимация 3	0,14	0,11	102	249	157	526	89	46

Конечно же, аппроксимация полиномом на некотором отрезке (z_1, z_3) всегда хуже аппроксимации полиномом того же порядка на двух отрезках (z_1, z_2) и (z_2, z_3). Однако в нашем случае, во-первых, зависимость на отрезках $(0, n_0)$ и $(n_0, 2L_k)$ значительно различаются (рис. 5), во-вторых, точка сочленения кривых находится в пределах $0{,}88\,L_k \ldots 1{,}22\,L_k$ для всех исследованных алгоритмов и компьютеров экспериментальной базы.

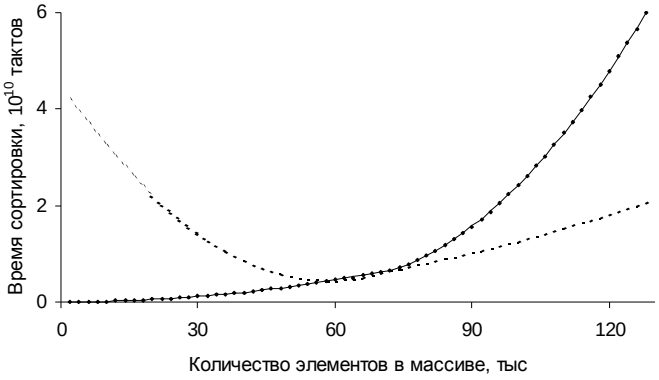

Рис. 5. Зависимость времени сортировки A_3 от размера массива

Кроме того, для некоторых алгоритмов вероятностные зависимости $t(n)$ являются квадратичными, и изменение зависимости в точке n_0 не предполагается. Это подтверждает качественное изменение $t(n)$ с появлением и ростом количества промахов в кэш при $n > n_0$.

На рис. 5 маркерами отмечены экспериментальные данные, сплошная линия – теоретическая зависимость вида (10). Пунктиром показаны экстраполяционные зависимости $t_1(n)$ на область $n > n_0$ и $t_2(n)$ на область $n < n_0$.

6. Выводы

Полученные экспериментальные данные и результаты дают возможность сделать следующие выводы:

1) семейство ЭВМ на базе процессора Intel и его аналогов является существенно нелинейно прогнозируемым по времени выполнения программ (рис. 1, 2);

2) вероятностных оценок вычислительной сложности недостаточно для принятия решений о выборе алгоритма из числа альтернативных для их включения в состав конкретного ПО;

3) изменение степени кэширования *PK* может существенно влиять на временную эффективность алгоритмов и программ. Областью такого снижения является $L_d > L_k$ (или $L_d > L_K - L_K^*$, где L_K^* – область кэша, занятая другими задачами), так как снижение *PK* при $L_d > L_k$ изменяет зависимость $t(n)$ b и при этом может наблюдаться существенное снижение временной эффективности;

4) степень влияния кэширования данных на временную эффективность алгоритмов различна. Есть алгоритмы, для которых изменение *PK* влечет значительное снижение временной эффективности. Есть также и алгоритмы, для которых такое изменение практически несущественно;

5) представленная методика позволяет качественно оценить влияние степени кэширования данных на временную эффективность алгоритмов;

6) для обоснованного принятия решений о выборе алгоритмов и рациональном подборе программно-аппаратных систем необходима количественная оценка влияния степени кэширования данных на временную эффективность алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. Шинкаренко В.И. Особенности оценки эффективности вычислительных алгоритмов // Проблемы программирования. –2001. – № 1–2. – С. 23–29.
2. Катленд Н. Вычислимость. Введение в теорию вычислимых функций. – М.: Мир, 1983. – 256 с.
3. Вирт Н. Алгоритмы + структуры данных = программа. – М.: Мир, 1985. – 406 с.
4. Кнут Д.Э. Искусство программирования. – Т. 1: Основные алгоритмы. – М.: Издательский дом "Вильямс", 2000. – 720 с.
5. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 2001. – 960 с.
6. Глибовець М.М. Основи комп'ютерних алгоритмів. – Київ: Видавничій дім "КМ Академія", 2003. – 452 с.
7. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов / Под ред. В.В. Мартынюка. – М.: Мир, 1981. – 366 с.
8. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536с.
9. Трауб Дж., Вожьняковский Х. Общая теория оптимальных алгоритмов. – М.: Мир, 1983. – 382 с.
10. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. – М.: Наука, Гл. ред. физ.-мат. лит., 1987. – 288 с.
11. Седжвик Р. Фундаментальные алгоритмы на С. Анализ / Структуры данных / Сортировка / Поиск / Алгоритмы на графах. – СПб.: ООО "ДиаСофтЮП", 2003. – 1136 с.
12. Макконелл Дж. Анализ алгоритмов. Вводный курс. – М.: Техносфера, 2002. – 304 с.
13. Шинкаренко В.И. Временная оценка операций обработки структурированных данных с учетом конвейеризации и кэширования // Проблемы программирования. – 2006. – № 2–3. – С. 43–52.
14. Касперски К. Техника оптимизации программ. Эффективное использование памяти. – СПб.: БХВ-Петербург, 2003. – 464 с.
15. Математическая энциклопедия / Гл. ред. Т.И. Виноградов. – Т. 4. – М.: Советская энциклопедия, 1984. – 1216 с.
16. Юров В. Assembler. – СПб.: Питер, 2001. – 624 с.