УДК 004.2

В.И. ШИНКАРЕНКО

СТАТИСТИЧЕСКИЕ ПОКАЗАТЕЛИ ЗАВИСИМОСТИ ВРЕМЕННОЙ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ ОТ КЭШИРОВАНИЯ ДАННЫХ

Abstract: *The problem of decreasing of time's efficiency of algorithms, when volume of processing data exceeds the size of cache memory is considered. That is the cache shortage effect. The statistical indexes for the estimation of this effect are offered. The advantages of these indexes are shown. The special cases of cache shortage effect are considered at processing different type's data.*
Key words: *algorithm, time's efficiency, caching data, cache shortage effect, statistical index.*

Анотація: *Розглядається проблема зниження часової ефективності алгоритмів при перевищенні обсягом даних, що обробляються, розміру кеш-пам'яті ЕОМ – ефекту нестатку кешу. Запропоновані статистичні показники оцінки цього ефекту. Наведені переваги даних показників над потенційними іншими. Розглянуті окремі випадки прояву ефекту нестатку кешу при обробці даних різних типів.*
Ключові слова: *алгоритм, часова ефективність, кешування, ефект нестатку кешу, статистичний показник.*

Аннотация: *Рассматривается проблема снижения временной эффективности алгоритмов при превышении объемом обрабатываемых данных размера кэш-памяти ЭВМ – эффекта недостатка кэша. Предложены статистические показатели оценки этого эффекта. Показаны преимущества данных показателей. Рассмотрены частные случаи проявления эффекта недостатка кэша при обработке данных разного типа.*
Ключевые слова: *алгоритм, временная эффективность, кэширование, эффект недостатка кэша, статистический показатель.*

1. Введение

Решение задач оценки эффективности алгоритмов способствует повышению качества разрабатываемого и эффективности использования разработанного программного обеспечения. Этой проблеме уделяется серьёзное внимание многими исследователями [1–6 и др.].

Одним из направлений таких исследований является оценка эффективности системы "алгоритм – исполнительное устройство".

В [7] исследовалась одна из частных задач – эффект снижения временной эффективности алгоритмов при объемах данных, обрабатываемых алгоритмами, превышающими размер кэша. Он связан с увеличением количества промахов в кэше [8] или (иначе) снижением степени кэширования [7]. Для краткости дальнейшего изложения назовем этот эффект эффектом недостатка кэша.

Для принятия обоснованного решения при подборе алгоритма или разработке нового, при выборе архитектуры ЭВМ с соответствующими модификациями под конкретное программное обеспечение необходимо наличие количественных показателей влияния архитектуры ЭВМ на эффективность алгоритма и методов их определения.

2. Качественные проявления эффекта недостатка кэша

На примере алгоритмов сортировки в [7] показано, что зависимость времени выполнения алгоритмов от количества элементов обрабатываемых данных $t(n)$ является кусочно-гладкой с изломом в точке, соответствующей размеру кэша. Для алгоритмов с вычислительной сложностью $O(n^2)$ и $O(n\ln(n))$ эти зависимости имеют вид (1) или (2) соответственно

$$t(n) = \begin{cases} a_1 n^2 + b_1 n + c_1 \text{ при } n <= n_0 \\ a_2 n^2 + b_2 n + c_2 \text{ при } n > n_0 \end{cases}; \quad (1)$$

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

где a_i, b_i, c_i – коэффициенты, n_0 – количество элементов массива, соответствующее точке изменения зависимости. Для случая однородных данных $n_0 \approx L_0 / l_Э$, где L_0 – объем данных, близкий к размеру кэша, $l_Э$ – размер элементов данных.

В общем виде для алгоритмов обработки однородных данных

$$t(n)=\begin{cases} f_1(n) \text{ при } n<=n_0 \\ f_2(n) \text{ при } n>n_0 \end{cases}, \quad (3)$$

для любых данных (однородных и неоднородных)

$$t(L_d)=\begin{cases} f_1(L_d) \text{ при } L_\text{d}<=L_0 \\ f_2(L_d) \text{ при } L_\text{d}>L_0 \end{cases}, \quad (4)$$

где L_d – объем обрабатываемых данных.

Эффект недостатка кэша проявляется в том, что при $L_\text{d} > L_0$ $f_2(L_d) > f_1(L_d)$. Задача заключается в количественной оценке снижения временной эффективности алгоритмов при $L_\text{d} > L_0$.

3. Методика численных экспериментов

Для определения показателей эффекта недостатка кэша необходимо знать зависимость вида (3). Параметры зависимости (3) зависят как от алгоритмов, так и от конструктивных особенностей ЭВМ. Используется статистический подход к их определению.

Рассмотрим методику численных экспериментов на примере алгоритмов сортировки. Они являются репрезентативными представителями алгоритмов обработки больших объемов данных. Во избежание разночтений и подробных спецификаций самих алгоритмов выбраны алгоритмы сортировки, приведенные Н. Виртом в [1], с идентичными текстами программ на ПАСКАЛе: A_1 – быстрая сортировка [1, с. 99]; A_2 – пирамидальная сортировка [1, с. 95]; A_3 – сортировка простым включением [1, с. 79]; A_4 – Шейкер-сортировка [1, с. 86]; A_5 – сортировка простым выбором [1, с. 82]; A_6 – сортировка пузырьком [1, с. 84] минимально модифицированная с отслеживанием момента завершения; A_7 – сортировка бинарным включением [1, с. 80]; A_8 – сортировка Шелла [1, с. 99] с рекомендованной Д. Кнутом последовательностью приращений 1, 3, 7, 15, 31.

Экспериментальная база с технической стороны состояла из 8 компьютеров различной конфигурации, различающихся объемом кэш-памяти, частотой процессора, шины и другими техническими характеристиками. Приведем размеры кэш-памяти данных уровней $L1$ и $L2$. На ЭВМ C_1, C_2 – 16/256, C_3 – 8/256, C_4, C_5 – 8/512, C_6 – 16/1024, C_7 – 64/64 и C_8 – 64/256 Kb (более подробная спецификация ЭВМ представлена в [7]).

Эксперименты выполнялись в MS DOS 6.22, что исключает 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.

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.

Экспериментальные значения $t(n)$ определялись в 64 точках с постоянным шагом n в интервале $(0, 2L_K)$, где L_K – размер кэша. При каждом n вычислялось среднее из 9 значений экспериментальных данных, полученных при различных начальных заполнениях массивов случайными числами. Параметры a_i, b_i, c_i и n_0 зависимости (1) или (2) определялись методом наименьших квадратов. Вид зависимости выбирался в соответствии с вычислительной сложностью алгоритма.

4. Количественная оценка эффекта недостатка кэша

Так как эффект недостатка кэша проявляется тем, что зависимость $t(n)$ в некоторой точке имеет излом, естественно, что количественные показатели должны учитывать степень излома.

Можно представить несколько таких показателей, например, изменение угла наклона кривой в точке n_0 (5):

$$\phi = \frac{t'(n_0)\big|n > n_0}{t'(n_0)\big|n < n_0}. \qquad (5)$$

Однако возникают сложности при их применении. Это и сложность интерпретации значений показателей, и сложность практического применения (необходимо знать значения, при которых эти показатели являются удовлетворительными [10]), а также чувствительности к статистическим результатам численных экспериментов.

Более удачными являются показатели, предложенные далее.

Учитывая тот факт, что размеры кэш-памяти составляют 128Кб, 256Кб, 512Кб, 1Мб, 2Мб, т.е. являются степенным рядом двойки, возникает практический вопрос: насколько улучшится временная эффективность алгоритмов при удвоении размеров кэша?

Предлагаются два следующих показателя:

– S_{K1} – насколько (на какую часть), в среднем, улучшится время выполнения алгоритма при удвоении кэша, если объем обрабатываемых данных находится в пределах $L_K < L_d < 2L_K$;

– S_{K2} – насколько минимально улучшится время выполнения алгоритма при удвоении кэша, если объем обрабатываемых данных превосходит $2L_K$.

Пусть статистическая зависимость времени выполнения алгоритма от объема обрабатываемых данных $t(n)$ имеет вид (3). Экстраполируем функцию $f_1(n)$ на отрезок $(L_K, 2L_K)$ (рис. 1).

Геометрическую интерпретацию предлагаемых показателей можно представить как

$$S_{K1} = \frac{S_1}{S_1 + S_2} \times 100\% = \frac{\int_m^{2m} (f_2(n) - f_1(n))dn}{\int_m^{2m} f_2(n)dn} \times 100\%; \quad (6)$$

$$S_{K2} = \frac{f_2(2L_K) - f_1(2L_K)}{f_2(2L_K)} \times 100\%, \quad (7)$$

где $m = L_K / l_Э$.

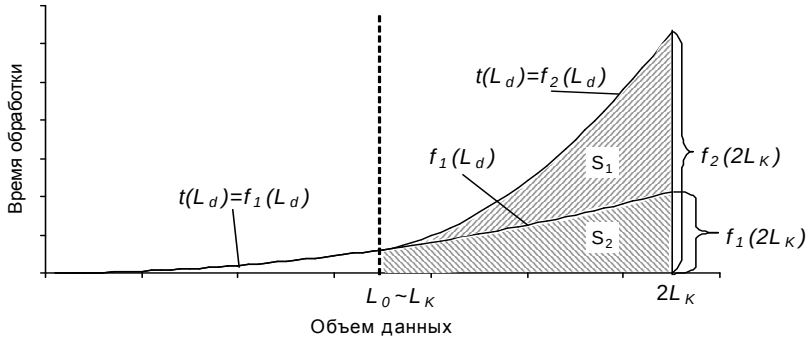

Рис. 1. Геометрическая интерпретация показателей S_{K1} и S_{K2}

Показатели S_{K1} и S_{K2} хорошо согласуются с предложенными ранее [9] показателями временной эффективности алгоритмов. В [9] рассматривается эксплуатационная характеристика функционально эквивалентных алгоритмов – степень превосходства одного алгоритма над другим. Предложена методика проведения численных экспериментов и расчета соответствующего показателя. Считая, что временная эффективность алгоритма есть функция

$$E = f(v, t, x, \psi, r), \quad (8)$$

где v – объем, t – тип, x – значения входных данных соответственно, ψ – программная среда функционирования и создания *.exe файла, r – архитектура ЭВМ. Переменные определены на множествах V^* – возможных значений объема данных, T^* – возможных типов данных, X^* – возможных значений данных, Ψ^* – возможных программных сред, $\Re^*$ – используемых множеств архитектур ЭВМ, на которых предполагается реализация алгоритмов.

Степень превосходства одного (i-го) алгоритма над другим (j-тым) на ограниченных множествах, являющихся подмножествами указанных выше множеств $\overline{V} \subseteq V^*$, $\overline{T} \subseteq T^*$, $\overline{X} \subseteq X^*$, $\overline{\Psi} \subseteq \Psi^*$, $\overline{\Re} \subseteq \Re^*$, есть

$$SUP_{ij}\left|\overline{V},\overline{T},\overline{X},\overline{\Psi},\overline{\Re}=\frac{1}{N}\sum_{v\subset\overline{V}}\sum_{t\subset\overline{T}}\sum_{x\subset\overline{X}}\sum_{\psi\subset\overline{\Psi}}\sum_{r\subset\overline{\Re}}\frac{f_j(v,t,x,\psi,r)-f_i(v,t,x,\psi,r)}{\max(f_i(v,t,x,\psi,r),f_j(v,t,x,\psi,r))}\right., \quad (9)$$

где N – общее количество слагаемых, f_i – время выполнения i-го алгоритма.

Показатели S_{K1} и S_{K2} являются частным случаем показателя SUP_{ij}. Для рассмотренных алгоритмов сортировки исследуемая область $\overline{\Omega}_{1i}=\overline{V}_1\times\overline{T}_1\times\overline{X}\times\overline{\Psi}_1\times\overline{\Re}_{1i}$ и вспомогательные $\overline{\Omega}_{2i}=\overline{V}_1\times\overline{T}_1\times\overline{X}\times\overline{\Psi}_1\times\overline{\Re}_{2i}$, $\overline{\Omega}_{3i}=\overline{V}_2\times\overline{T}_1\times\overline{X}\times\overline{\Psi}_1\times\overline{\Re}_{1i}$ и $\overline{\Omega}_{4i}=\overline{V}_2\times\overline{T}_1\times\overline{X}\times\overline{\Psi}_1\times\overline{\Re}_{2i}$, где $\overline{V}_1=(L_K,2L_K)$, $\overline{V}_2=\{2L_K\}$, $\overline{T}_1=\{\text{integer}\}$, $\overline{X}$ – массивы случайных чисел в диапазоне допустимых значений для типа данных, $\overline{\Psi}_1$={MS DOS 6.22, Delphi 7 с оптимизацией кода, WDOSX – программа конвертирования EXE-модуля в формат DOS}, $\overline{\Re}_{1i}=\{C_i\}$ – один из компьютеров экспериментальной базы, $\overline{\Re}_{2i}=\{C_i$ с удвоенным кэшем}.

Показатели S_{K1} и S_{K2} можно представить следующим образом:

$$S_{K1}(A)\left|\overline{\Omega}_{1i}=SUP_{ij}\right|\begin{matrix}A_i=A\left|\overline{\Omega}_{1i}\right.\\A_j=A\left|\overline{\Omega}_{2i}\right.\end{matrix}, \quad (10)$$

$$S_{K2}(A)\left|\overline{\Omega}_{1i}=SUP_{ij}\right|\begin{matrix}A_i=A\left|\overline{\Omega}_{3i}\right.\\A_j=A\left|\overline{\Omega}_{4i}\right.\end{matrix}. \quad (11)$$

Естественно, показатели S_{K1} и S_{K2} могут быть определены на другом множестве Ω.

Показатели S_{K1} и S_{K2} имеют преимущество перед другими (например, (5)) в том, что они:

– имеют простую графическую интерпретацию;

– дают возможность оценки допустимых значений при принятии решений;

– хорошо согласуются с системой статистических показателей временной эффективности алгоритмов.

Значения показателей S_{K1} и S_{K2} для алгоритмов $A_1\ldots A_8$ в области Ω_{1i} приведены в табл. 1.

Таблица 1. Показатели S_{K1}/S_{K2} для алгоритмов $A_1\ldots A_8$ в области Ω_{1i}

		ЭВМ из экспериментальной базы								
		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	Среднее
Алгоритм сортировки	A_1	4/5	2/3	1/1	1/1	1/1	0/0	1/1	2/3	2/2
	A_2	6/5	11/13	8/12	5/8	11/16	0/0	10/10	8/10	7/9
	A_3	36/50	53/67	26/38	13/21	39/54	6/10	46/59	12/25	29/41
	A_4	21/32	33/46	4/6	1/2	9/14	0/0	24/33	4/9	12/18
	A_5	50/58	46/54	7/10	7/9	14/18	3/5	44/52	36/52	26/32
	A_6	42/50	52/60	5/6	3/4	14/17	1/1	24/30	7/12	19/23
	A_7	49/63	51/67	37/51	23/34	50/64	18/29	49/63	15/29	37/50
	A_8	26/39	57/71	82/90	66/77	88/93	64/76	69/79	40/49	62/72
	Среднее	29/38	38/48	21/27	15/20	28/35	12/15	33/41	16/24	24/31

Качественные заключения по проведенным численным экспериментам [7] о том, что, во-первых, различные алгоритмы имеют различную “чувствительность” к изменению размера кэша и, во-вторых, компьютеры различной архитектуры и модификации в различной степени способны изменять временную эффективность совокупности алгоритмов, нашли свое количественное воплощение.

Анализ данных табл. 1 показывает:

– со стороны архитектуры компьютеров – для ЭВМ с большим размером кэша показатели S_{K1} и S_{K2} (табл. 2) ниже, т.е. удвоение размеров кэша в этом случае менее эффективно. Теоретически этого следовало ожидать, так как при увеличении размера кэша больше данных находится в нем и вероятность промаха снижается. С другой стороны эту зависимость можно рассматривать лишь как тенденцию. Как контрпример можно привести результаты по ЭВМ C_2 (L_K =256Кб), где показатели хуже, чем на ЭВМ C_7 (L_K =128Кб);

– со стороны алгоритмов – можно отметить, что алгоритмы по-разному реагируют на удвоение кэша: некоторые алгоритмы значительно ускоряются (A_7, A_8), некоторые практически на такое изменение не реагируют (A_1, A_2).

Таблица 2. Средние показатели S_{K1} и S_{K2} по ЭВМ с одинаковым размером кэш-памяти

	Показатель	Размер кэш-памяти			
		128Кб	256Кб	512Кб	1024Кб
В среднем по алгоритмам A_1 … A_8	S_{K1}	33	26	22	12
	S_{K2}	41	34	27	15
По алгоритму A_8	S_{K1}	69	51	77	64
	S_{K2}	79	62	85	76

Показатели S_{K1} и S_{K2} построены на основе статистических данных. Возникает естественный вопрос о точности таких показателей.

Выборочно выполнено более 100 повторных (идентичных) экспериментов с последующим расчетом S_{K1} и S_{K2}. На полученной выборке в 90% случаев расхождения показателей были в пределах 1% и лишь в единичных случаях немногим превышали 5%. Естественно, увеличение количества измерений приведет к повышению точности показателей.

Отметим, что вопрос устойчивости таких измерений требует отдельных исследований, особенно это актуально для неустойчивых сред функционирования программ и алгоритмов – многозадачных ОС.

5. Проявление эффекта недостатка кэша для алгоритмов с повышенной долей операций обработки данных

Алгоритмы не полностью привязаны к типам данных. Так, алгоритмы сортировки способны отсортировать любую последовательность объектов, принадлежащих множеству с заданным отношением порядка: числа, полиномы, матрицы и т.п., при этом изменится лишь операция

сравнения. Так как S_{K1} и S_{K2} зависят от типов данных, для их определения в каждом случае необходимо проводить отдельные численные эксперименты.

Представленные в табл. 1 результаты соответствуют обработке массивов целых чисел. Возникает вопрос: а как проявится эффект недостатка кэша при изменении типа данных? Он будет усиливаться или ослабевать и насколько?

Представим среднестатистическое время выполнения алгоритма как

$$t(n) = \begin{cases} f_1(n) = \alpha_1(n) + \beta_1(n), & n \in (0, m) \\ f_2(n) = \alpha_2(n) + \beta_2(n) & n \in (m, 2m) \end{cases}, \tag{12}$$

где α_i – время на выполнение операций обмена данными, β_i – время на операции обработки данных, $m = L_K / l_Э$. Отметим, что зависимость (12) – приближенная не только в силу статистического характера. Время выполнения операций с учетом кэширования и конвейеризации не постоянно и в значительной степени зависит от предыстории выполненных операций.

Рассмотрим частные случаи изменения типа данных, обрабатываемых алгоритмом:

– тип данных изменяется с сохранением размера;

– тип данных изменяется только в части размера (операции обработки данных остаются те же и в том же количестве);

– при изменении типа данных изменятся их размер и операции обработки данных.

Рассмотрим первый случай по схеме: гипотезы об изменении процесса вычислений и показателей S_{K1} и S_{K2} $\rightarrow$ теоретическая проверка гипотезы о изменении показателей $\rightarrow$ экспериментальное её подтверждение.

Допустим, операции обработки данных будут более длительными. Потери от промахов в кэш при пересылке данных могут в большей степени компенсироваться параллельным выполнением более длительных операций обработки данных. Можно предположить, что показатели S_{K1} и S_{K2} будут уменьшаться.

Обоснуем это положение теоретически. Пусть тип данных T_1 заменен типом данных T_2 ($T_1 \rightarrow T_2$). При этом размер элементов данных T_1 и T_2 одинаков, а множества допустимых значений – разные, причем операции обработки данных типа T_2 более длительны.

Оценим разницу $\Delta S_{K1}\big|_{T_1 \rightarrow T_2} = S_{K1}\big|_{T_1} - S_{K1}\big|_{T_2}$.

Обозначим $F_{iTj} = \int_m^{2m} f_i(n)\Big|_{T_j} dn$, $A_{iTj} = \int_m^{2m} \alpha_i(n)\Big|_{T_j} dn$ и $B_{iTj} = \int_m^{2m} \beta_i(n)\Big|_{T_j} dn$.

Согласно (6) получаем

$$\Delta S_{K1}\big|_{T_1 \rightarrow T_2} = \frac{F_{2T1} - F_{1T1}}{F_{2T1}} - \frac{F_{2T2} - F_{1T2}}{F_{2T2}}. \tag{13}$$

Чтобы показать, что $\Delta S_{K1}\big|_{T_1 \rightarrow T_2} > 0$, достаточно показать, что $F_{2T2} > F_{2T1}$ и $F_{2T2} - F_{1T2} < F_{2T1} - F_{1T1}$ (числитель второй дроби меньше, а знаменатель – больше).

Преобразуем первое неравенство с учетом (12): $A_{2T2} + B_{2T2} > A_{2T1} + B_{2T1}$ и $B_{2T2} - B_{2T1} > A_{2T1} - A_{2T2}$. Истинность последнего следует из того, что снижение времени обмена данными за счет распараллеливания обмена с обработкой возможно лишь в пределах увеличения времени обработки данных.

Второе неравенство преобразуем к виду

$$A_{2T2} + B_{2T2} + A_{1T1} + B_{1T1} < A_{1T2} + B_{1T2} + A_{2T1} + B_{2T1}.$$

Оно истинно ввиду того, что $A_{2T1} > A_{2T2}$ – за счет большего распараллеливания операций обмена и обработки данных; $A_{1T1} = A_{1T2}$ ($\alpha_1(n)\big|_{T_1} = \alpha_1(n)\big|_{T_2}$) – количество операций обмена при обоих типах данных одинаково, недостатка кэша нет, значит, и время на операции обмена – одинаково; $B_{1T1} = B_{2T1}$ ($\beta_1(n)\big|_{T_1} = \beta_2(n)\big|_{T_1}$) и $B_{1T2} = B_{2T2}$ ($\beta_1(n)\big|_{T_2} = \beta_2(n)\big|_{T_2}$) – зависимость количества операций обработки данных не изменяется на отрезке $(0, 2L_K)$ и интеграл в одних пределах будет одинаковым.

Аналогично можно показать, что $\Delta S_{K2}\big|_{T_1 \to T_2} > 0$.

Для алгоритмов сортировки этот случай соответствует, например, замене типа данных integer на тип данных single (4-байтовые вещественные числа) или int64 на double (8-байтовые целые и вещественные числа) в Delphi7.

Выполненные численные эксперименты действительно подтверждают тенденцию к уменьшению S_{K1} и S_{K2} при увеличении длительности операций обработки данных.

Усредненные значения S_{K1} и S_{K2} при типах данных $T_1 \ldots T_4$:

$$\overline{S}_{Km}(T_k) = \frac{1}{N_A N_C} \sum_{i=1}^{N_A} \sum_{j=1}^{N_C} S_{Km}(A_i)\Big|_{\overline{\overline{\Omega}}_{jk}}, \qquad (14)$$

где $\overline{\overline{\Omega}}_{jk} = \overline{V_1} \times T_K \times \overline{X} \times \overline{\Psi_1} \times \{C_j\}$, N_A – количество алгоритмов (N_A=8), N_C – количество ЭВМ экспериментальной базы (N_C=8) и $m = 1,2$, приведены в табл. 3.

В табл. 3 указаны также средние значения по алгоритмам и компьютерам:

$$\tilde{S}_{Km}(A_i) = \frac{1}{N_T N_C} \sum_{k=1}^{N_T} \sum_{j=1}^{N_C} S_{Km}(A_i)\Big|_{\overline{\overline{\Omega}}_{jk}}, \qquad (15)$$

где N_T – количество типов данных (N_T =4) и

$$\hat{S}_{Km}(C_j) = \frac{1}{N_A N_T} \sum_{i=1}^{N_A} \sum_{k=1}^{N_T} S_{Km}(A_i)\Big|_{\overline{\overline{\Omega}}_{jk}}. \qquad (16)$$

Отметим высокую корреляцию результатов экспериментов, приведенных в табл. 1 и 3.

Пример зависимости времени сортировки от объема данных алгоритма A_5 для ЭВМ C_1 дан на рис. 2. Степень излома кривых T_1 и T_3 больше, чем T_2 и T_4 соответственно.

Таблица 3. Средние значения S_{K1} и S_{K2} при изменении типов данных с элементами одинакового размера

Тип данных	T_1={integer}		T_2 ={single}		T_3 ={int64}		T_4 ={double}	
$\overline{S}_{K1} / \overline{S}_{K2}$	27/31		13/17		17/23		15/20	
Алгоритм	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
$\tilde{\tilde{S}}_{K1} / \tilde{\tilde{S}}_{K2}$	1/2	6/9	16/23	7/11	20/24	13/14	23/32	45/58
ЭВМ	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
$\hat{S}_{K1} / \hat{S}_{K2}$	25/33	35/45	11/14	7/10	15/19	6/9	22/28	10/14

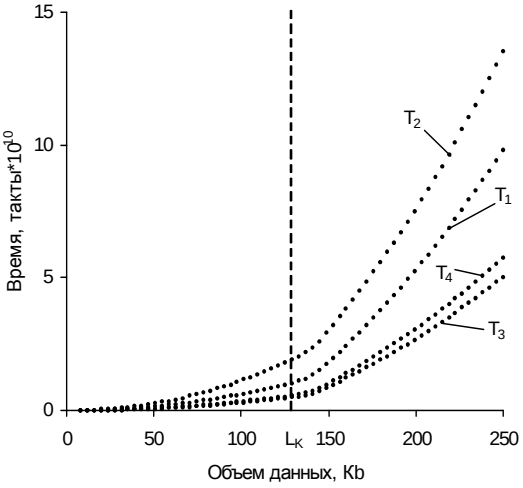

Рис. 2. Зависимость времени сортировки от объема данных типа $T_1 ... T_4$

6. Проявление эффекта недостатка кэша для алгоритмов с повышенной долей операций обмена данными

Рассмотрим случай, когда с изменением типа данных увеличивается размер элементов, а операции обработки данных остаются теми же ($T_1 \to T_2$). В этом случае увеличивается “ценой” промахов в кэш: при каждом промахе в кэш будет загружаться большая порция данных. Следует ожидать роста показателей S_{K1} и S_{K2}.

Посмотрим теоретически и экспериментально, что показатели S_{K1} и S_{K2} действительно растут.

Изменение показателя $\Delta S_{K1}\big|_{T_1 \to T_2}$ можно представить так же, как в (13), с тем отличием, что $F_{iTj} = \int_{m_j}^{2m_j} f_i(n)\Big|_{T_j} dn$, где $m_j = L_K / l_{Эj}$, $l_{Эj}$ – размер элементов типа T_j. Аналогично изменятся пределы интегрирования A_{iTj} и B_{iTj}.

Так как $B_{1T1} = B_{2T1}$, $B_{1T2} = B_{2T2}$, $A_{1T1} = A_{1T2}$ по указанным в предыдущем пункте причинам, преобразуем (13) к виду

$$\Delta S_{K1}\big|_{T_1 \to T_2} = \frac{A_{2T1} - A_{1T1}}{A_{2T1} + B_{2T1}} - \frac{A_{2T2} - A_{1T1}}{A_{2T2} + B_{2T2}}. \qquad (17)$$

Обозначив $p = A_{2T2} - A_{1T1}$, $q = A_{2T2} + B_{2T2}$, $\Delta p = A_{2T1} - A_{2T2}$ и $\Delta q = B_{2T1} - B_{2T2}$ определим знак ΔS как

$$sign(\Delta S_{K1}\big|_{T_1 \to T_2}) = sign(\frac{p + \Delta p}{q + \Delta q + \Delta p} - \frac{p}{q});$$

$$sign((p+\Delta p)q - p(q+\Delta q+\Delta p)) = sign(\Delta p(q-p) - \Delta qp).$$

Учитывая, что $A_{2T1} > A_{1T1}$ – в силу эффекта недостатка кэша, $A_{2T1} < A_{2T2}$ – объем данных, подлежащих пересылке, тот же, а "цена" промахов в кэш выше во втором случае; $B_{1T1} > B_{1T2}$ и $B_{2T1} > B_{2T2}$ – функция $\beta(n)$ одинаковая и монотонно возрастающая, а интеграл берется на промежутке, смещенном к нулю ($m_2 < m_1$ и $2m_2 < 2m_1$) и меньшего размера ($2m_2 - m_2 < 2m_1 - m_1$). Получаем $sign(q-p) = sign(\Delta q) = sign(p) = 1$ и $sign(\Delta p) = -1$, откуда $sign(\Delta S_{K1}|_{T_1 \to T_2}) = -1$ т.е. $\Delta S_{K1}|_{T_1 \to T_2} < 0$.

Выполнен соответствующий эксперимент. Для алгоритмов сортировки исследовались следующие типы данных:

```
T5=record      T6=record                T7=record                    T8=record
  x:integer;     x:integer;               x:integer;                   x:integer;
end;             y:array[1..3] of integer;  y:array[1..15] of integer;  y:array[1..31] of integer;
               end;                     end;                         end.
```

Операции обработки в алгоритмах сортировки – операции сравнения по полю x:integer (одного типа) – абсолютно одинаковые для типов $T_5 \ldots T_8$, а операции обмена – пересылка записей разного размера.

В табл. 4 приведены средние значения S_{K1} и S_{K2} аналогично (14) – (16) с той разницей, что $\overline{\Psi}_1$ заменяется на $\overline{\Psi}_2$={MS DOS 6.22, Delphi 7 без оптимизации кода, WDOSX – программа конвертирования EXE-модуля в формат DOS}.

Приведенные результаты хорошо согласуются с теоретическим прогнозом.

Таблица 4. Средние значения S_{K1} и S_{K2} при изменении типов данных с элементами переменного размера

Тип данных	T_5 (record 4b)		T_6 (record 16b)		T_7 (record 64b)		T_8 (record 128b)	
$\overline{S}_{K1} / \overline{S}_{K2}$	14/18		19/25		26/33		30/38	
Алгоритм	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
$\tilde{S}_{K1} / \tilde{S}_{K2}$	3/4	9/14	22/31	10/15	50/56	20/24	24/34	39/53
ЭВМ	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
$\hat{S}_{K1} / \hat{S}_{K2}$	29/37	35/44	18/24	14/19	25/32	12/16	26/33	19/25

Отметим, что типы T_1 и T_5 практически идентичны в том плане, что транслятор Delphi 7 для операций пересылки и сравнения строит абсолютно идентичные коды. Отличие данных, приведенных в табл. 3 и 4, состоит в оптимизации исследуемого кода во втором случае.

Согласно проведенным экспериментам, эффект недостатка кэша усиливается при оптимизации кода. По-видимому, при сокращении времени обработки данных (в данном случае при удалении излишних команд) загрузка данных в кэш в меньшей степени распараллеливается с обработкой данных.

Скорее всего, эта закономерность должна проявляться в большинстве случаев для алгоритмов обработки больших объемов данных. Однако теоретически подтвердить это не удалось, а целенаправленные эксперименты не проводились.

На основании результатов первых двух случаев прогнозы относительно изменения показателей S_{K1} и S_{K2} можно делать в случаях:

– снижения длительности операций обработки данных и увеличения размеров данных;

– повышения длительности операций обработки данных и уменьшения размеров данных.

В остальных двух случаях необходимо проведение численных экспериментов.

7. Выводы

Практическое использование полученных результатов представляется в двух направлениях. Во-первых, для выполнения алгоритмов обработки больших объемов данных они позволяют подбирать соответствующую конфигурацию компьютера, что существенно для прикладных программ, где такие алгоритмы превалируют и повышены требования к их временной эффективности. В основном это касается объема кэш-памяти. В частности, подбор конфигурации компьютера может осуществляться при распределении задач в гетерогенных вычислительных сетях. Во-вторых, для подбора алгоритма с учетом конфигурации компьютера или адаптации алгоритмов к техническим средствам.

О значении эффекта недостатка кэша говорит тот факт, что при обработке сравнительно небольших объемов данных (порядка нескольких мегабайт) удвоение кэша в некоторых случаях может позволить снизить время выполнения алгоритмов в 4 … 4,5 раза ($S_{K1} \approx 80\ldots90\%$).

Выполненные исследования эффекта недостатка кэша при различных типах данных позволяют избегать дополнительных исследований при известных S_{K1} и S_{K2} для некоторого типа данных. Кроме того, эти исследования, согласуясь с теоретическими результатами, подтверждают достоверность оценок показателей S_{K1} и S_{K2}.

Представляется необходимым продолжение исследований эффекта недостатка кэша в менее стабильных средах функционирования алгоритмов – многозадачных операционных системах, таких как Windows, Unix, что связано с их доминирующим использованием в качестве системного программного обеспечения.

СПИСОК ЛИТЕРАТУРЫ

1. Кнут Д.Э. Искусство программирования. – М.: Издательский дом «Вильямс», 2000. – Т. 1: Основные алгоритмы. – 720 с.
2. Вирт Н. Алгоритмы + структуры данных = программа. – М.: Мир, 1985. – 406 с.
3. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М.: МЦНМО, 2001. – 960 с.
4. Макконелл Дж. Анализ алгоритмов. Вводный курс. – М.: Техносфера, 2002. – 304 с.
5. Глибовець М.М. Основи комп'ютерних алгоритмів. – Київ: Видавничій дім «КМ Академия», 2003. – 452 с.
6. Ахо А. Структуры данных и алгоритмы / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М.: Издательский дом «Вильямс», 2000. – 384 с.
7. Шинкаренко В.И. Зависимость временной эффективности алгоритмов и программ обработки больших объемов данных от их кэширования // Математичні машини і системи. – 2007. – № 2. – С. 43–55.

8. Касперски К. Техника оптимизации программ. Эффективное использование памяти. – СПб.: БХВ-Петербург, 2003. – 464 с.
9. Шинкаренко В.И. Сравнительный анализ временной эффективности функционально эквивалентных алгоритмов // Проблемы программирования. – 2001. – № 3–4. – С. 31–39.
10. Андон Ф.И. Основы инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Коротун и др. – К.: Академпериодика, 2007. – 670 с.

Стаття надійшла до редакції 12.09.2007