3. Холл, Марти, Браун, Лэрри. Программирование для Web. Библиотека профессионала /Пер. с англ. – М.: Вильямс, 2002. -1264с.: ил.
4. Client-Side JavaScript Guide.// Ch 15. Live Connect Overwiew.

Поступила в редакцию 15 мая 2006 г.

УДК 621.391.251

ДИНАМИЧЕСКОЕ АДРЕСНО-ВЕКТОРНОЕ СЖАТИЕ ДВОИЧНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

И.А.Кулик, доц.; С.Н. Харченко, магистр
Сумский государственный университет

С целью повышения коэффициента адресно-векторного сжатия более перспективным является применение динамических алгоритмов, которые позволяют кодеру быстрее, чем при статических алгоритмах, адаптироваться к изменениям в текущем потоке данных.

ВВЕДЕНИЕ

Для повышения эффективности использования коммуникационных и информационно-вычислительных ресурсов применяются различные методы и средства. Среди них важную роль играет сжатие, которое стало неотъемлемой частью систем хранения и передачи данных, обеспечивая необходимую скорость их обработки путем выявления скрытых резервов в производительности отдельных элементов и системы в целом, неучтенных на этапе проектирования. Это эквивалентно повышению пропускных способностей каналов передачи или увеличению емкости запоминающих устройств [1, 2].

Современный этап развития информационных систем характеризуется существенным ростом объемов передаваемых и обрабатываемых данных, которые различаются между собой назначением и форматом представления. В структуре информационных потоков доминирующее положение занимают мультимедийные данные, данные смешанного типа – текст, графика, видео и т.д. Следовательно, источники информации различного вида имеют нестационарный характер и могут изменять свои параметры в течение передачи не только информационного пакета, но и блока данных. Множество методов и алгоритмов сжатия настраиваются на фиксированные значения или, в лучшем случае, узкие области изменения параметров информационного источника (например, вероятностей появления двоичных слов). В случае же изменения характеристик источника или выхода их значений за предопределенные граничные значения такие методы сжатия теряют свою эффективность [2, 3].

ПОСТАНОВКА ЗАДАЧИ

Достойное место среди методов сжимающего кодирования занимает адресно-векторный метод, отличающийся высокой скоростью преобразования данных и простотой реализации [4, 5, 6]. Адресно-векторное кодирование заключается в переходе от векторного к адресному методу кодирования в зависимости от числа k двоичных единиц в n-разрядном слове. Такой переход осуществляется согласно системе неравенств:

$$\begin{cases} 0 \le k < \alpha - 1, \\ n - \alpha + 1 < k \le n, \end{cases} \tag{1}$$

где $\alpha = \dfrac{n}{\lceil \log_2 n \rceil}$.

В вероятностном виде система (1) неравенств – условий сжатия – представляется следующим образом [6]:

$$\begin{cases} 0 \le p < \dfrac{1}{\lceil \log_2 n \rceil} - \dfrac{1}{n} + \dfrac{\gamma}{n}, \\ 1 - \dfrac{1}{\lceil \log_2 n \rceil} + \dfrac{1}{n} - \dfrac{\gamma}{n} < p \le 1, \end{cases} \tag{2}$$

где p – вероятность появления двоичной единицы;

γ – допустимое заданное отклонение случайной величины k единиц от математического ожидания.

Из системы (2) следует, что метод адресно-векторного сжатия также является "чувствительным" к изменению параметров источника информации, генерирующего сжимаемые двоичные данные. Выход значения p за указанные области (2) снижает коэффициент сжатия до единицы [5, 6]. Существующие на сегодняшний день алгоритмы адресно-векторного сжатия являются статичными с точки зрения граничных значений условий сжатия (1, 2), что ограничивает их применение.

Таким образом, актуальной является задача разработки динамических процедур и на их основе динамических алгоритмов адресно-векторного сжатия, которые обладали бы способностью подстраиваться под изменяющийся источник информации. Как показывают (1, 2), одним из реальных способов для адаптации алгоритмов адресно-векторного сжатия является изменение длины n сжимаемых последовательностей.

То, что такая задача имеет реальный практический смысл, показывают следующие примеры.

Пример 1 На вход кодирующего устройства поступает двоичная 32-разрядная последовательность:

$$10000001000000001111011111011111. \tag{3}$$

Очевидно, что данная комбинация не сжимается адресно-векторным методом, так как при $n = 32$ система (1) условий сжатия выглядит следующим образом:

$$\begin{cases} 0 \le k < 5,4, \\ 26,6 < k \le 32, \end{cases} \tag{4}$$

а последовательность (3) содержит $k = 16$ двоичных единиц. Но разбивая (3) на две части по 16 разрядов, как показано на рисунке 1, и применяя к каждой из частей адресно-векторный метод кодирования согласно условиям (1) при $n = 16$:

$$\begin{cases} 0 \le k < 3, \\ 13 < k \le 16, \end{cases} \tag{5}$$

наблюдаем экономию двоичных разрядов приблизительно в 1,3 раза.

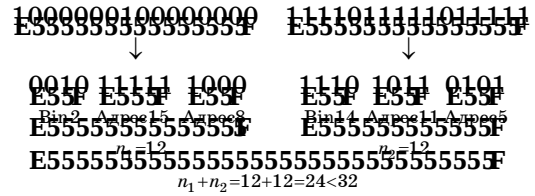

Рисунок 1 – Разбиение двоичной последовательности для динамического адресно-векторного сжатия

Пример 2 На вход кодирующего устройства поступает одна за одной следующие двоичные 16-разрядные последовательности:

$$1000000100001000\,, \tag{6}$$

$$0000000000000000\,. \tag{7}$$

При статическом подходе из системы условий (5) следует, что комбинация (6) не сжимается, а комбинация (7) сжимается адресно-векторным методом (рисунок 2).

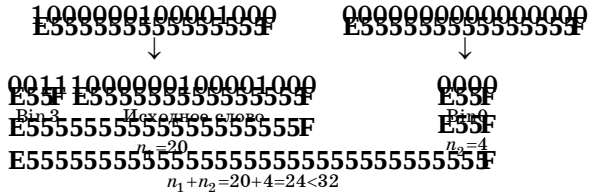

Рисунок 2 – Статическое адресно-векторное сжатие для двух 16-разрядных последовательностей

Теперь представим две двоичные комбинации (6) и (7) как одну 32-разрядную последовательность. Тогда согласно граничным условиям (4) адресно-векторное сжатие можно осуществить так, как показано на рисунке 3.

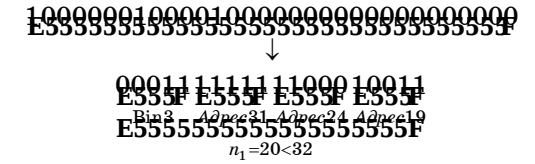

Рисунок 3 – Конкатенация 16-разрядных последовательностей для динамического адресно-векторного сжатия

В результате при статическом подходе (рисунок 2) коэффициент адресно-векторного сжатия составляет $32/24 \approx 1,3$, а при динамическом (рисунок 3) – $32/20 = 1,6$. Таким образом, для данного случая эффективность динамического подхода к адресно-векторному сжатию выше в $1,6/1,3 \approx 1,2$ раза.

РАЗРАБОТКА ДИНАМИЧЕСКОГО АЛГОРИТМА АДРЕСНО-ВЕКТОРНОГО СЖАТИЯ

Динамический подход к адресно-векторному сжатию, несмотря на большие аппаратурные затраты по сравнению со статическим, позволяет в большей степени устранять информационную избыточность при

нестационарности источников данных. В процессе динамического адресно-векторного сжатия должен одновременно осуществляться выбор оптимальной с точки зрения степени сжатия длины разбиения (конкатенации) последовательности данных. Таким образом, система адресно-векторного сжатия сразу приспосабливается под текущий поток данных для получения наилучшего коэффициента сжатия. При разработке динамических адресно-векторных алгоритмов необходимо сохранить одно из основных преимуществ рассматриваемого метода – его высокое быстродействие.

Предлагаемый в данной работе динамический алгоритм адресно-векторного сжатия принимает и обрабатывает информационные пакеты длиной 32 разряда. В случае невыполнения условий (1) возможно разбиение пакета на отдельные части размером 8, 16 или 24 разряда, для которых условия (1) сжатия будут выполняться. За счет возможности сжатия отдельных частей пакетов размер выходных данных все же уменьшится.

При разбиении 32-разрядного пакета на части с размером, кратным 8, необходим анализ каждого из них по отдельности. После этого рассматривается возможность объединения нескольких таких частей в последовательно расположенные комбинации длиной по 8, 16, 24 или 32 разряда с условием получения максимального сжатия при их дальнейшей обработке. Необработанные разряды рассматриваются в совокупности с новопришедшим на обработку следующим 32-разрядным пакетом.

Анализ частей информационного пакета на возможность их объединения производится на основании следующего алгоритма:

1 Для начала цикла обработки данных принимаются первые n разрядов из входного потока данных.

2 На следующем этапе запускается цикл обработки и получаются следующие m разрядов.

3 Производится проверка на сжимаемость первых n разрядов. В случае, если они сжимаемы, осуществляется переход к пункту 4, в противном случае – к пункту 5.

4 На этом этапе производится проверка сжимаемости следующих m разрядов. Если они также сжимаемы, то осуществляется переход к пункту 7, в противном случае – к пункту 6.

5 Если следующие m разрядов являются сжимаемыми, то осуществляется переход к пункту 6, в противном случае – к пункту 8.

6 Производится проверка сжимаемости совокупности $(n+m)$ разрядов. Если они сжимаемы, то осуществляется переход к пункту 9, в противном случае – к пункту 8.

7 Если первые n разрядов и следующие за ними m разрядов являются сжимаемыми, то выполняется проверка условия: коэффициент сжатия совокупности комбинаций из $(n+m)$ разрядов больше, чем коэффициент сжатия последовательности из m разрядов? Если данное условие выполняется, то осуществляется переход к пункту 8, в противном случае – к пункту 9.

8 Две последовательности из n и m разрядов объединяются в одну и $n = n + m$. Цикл обработки повторяется с переходом к пункту 2.

9 На этом этапе происходит адресно-векторное сжатие кодовой комбинации из n разрядов.

10 Выдача сжатой адресно-векторной комбинации.

11 В качестве первой комбинации используется следующая полученная $n = m$, и цикл обработки повторяется.

В приведенном алгоритме n – число разрядов первого кодового слова, которое формируется для дальнейшего адресно-векторного сжатия, его длина может варьироваться от 8 до 32 разрядов, m – число разрядов второго кодового слова, которое подается на обработку, его длина фиксирована 8 разрядами.

Каждая обработанная комбинация сопровождается заголовком, который содержит служебную информацию. Размер обработанной кодовой комбинации определяется битами A_1A_2, которые соответственно равны: 00 – 8 разрядов, 01 – 16 разрядов, 10 – 24 разряда, 11 – 32 разряда. Второй частью $A_3...A_7$ заголовка являются данные о количестве единиц в обрабатываемой последовательности. В зависимости от размера комбинации длина этой части изменяется от четырех до шести разрядов. Таким образом, формат выходных последовательностей при динамическом адресно-векторном сжатии будет иметь вид на рисунке 4.

A_1A_2	$A_3...A_7$	Данные

Рисунок 4 – Формат выходной последовательности данных

АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ ИССЛЕДОВАНИЯ

Для проведения сравнительной оценки статического и динамического алгоритмов адресно-векторного сжатия были проведены:

- кодирование с фиксированной длиной пакета 8 разрядов;
- кодирование с фиксированной длиной пакета 16 разрядов;
- кодирование с фиксированной длиной пакета 32 разряда;
- кодирование с переменной (кратной 8) длиной пакета (динамическое сжатие).

Для оценки эффективности динамического алгоритма адресно-векторного сжатия была разработана программная модель на языке высокого уровня. В качестве источника данных были использованы файлы с различным процентным содержанием двоичных единиц, группами по восемь и четыре разряда в однобайтном блоке (второй случай рассматривает вариант, когда блок может быть заполнен единицами только наполовину, что, с точки зрения (1), не дает сжатия вообще). При этом в обрабатываемом потоке эти информационные пакеты располагаются в произвольном порядке. Это достигается с помощью произвольного заполнения черного экрана размером 640×480 пикселей белыми и серыми точками, процентное содержание которых варьировалось от 0 до 100% с шагом в 5%. Получаемые таким образом данные впоследствии формировали файлы формата bmp.

На следующем этапе исследования полученные файлы подвергались адресно-векторному сжатию статическим алгоритмом при длине n, равной 8, 16 и 32 разряда, а также динамическим алгоритмом при переменной длине пакета. Результаты проведенных экспериментов для более наглядного восприятия представлены в графической форме (рисунки 5, 6). На приведенных графиках по оси абсцисс откладывается процентное содержание блоков с восьми- и четырехразрядными группами двоичных единиц в исследуемом файле, а по оси ординат – полученный в результате адресно-векторного кодирования коэффициент сжатия.

На основании анализа графиков можно выделить области, для которых динамический алгоритм адресно-векторного сжатия дает лучший результат. Например, согласно графику на рисунке 5 динамический алгоритм имеет преимущество над статическим сжатием при $n = 8$ и $n = 16$ в интервале от 0 до 10%. Так как максимальная длина пакета при

динамическом адресно-векторном сжатии равна 32 разряда, то графики для статического при $n = 32$ и динамического сжатия на начальных файлах должны совпадать. Но на практике между этими графиками отмечается расхождение, которое объясняется тем, что динамический алгоритм адресно-векторного сжатия генерирует два дополнительных разряда в заголовке. Причем сам заголовок соизмерим с длиной обрабатываемого пакета, поэтому динамический подход к адресно-векторному сжатию будет давать большую эффективность при таком размере блока, когда размер заголовка по сравнению с ним будет ничтожно малым и им можно будет пренебречь.

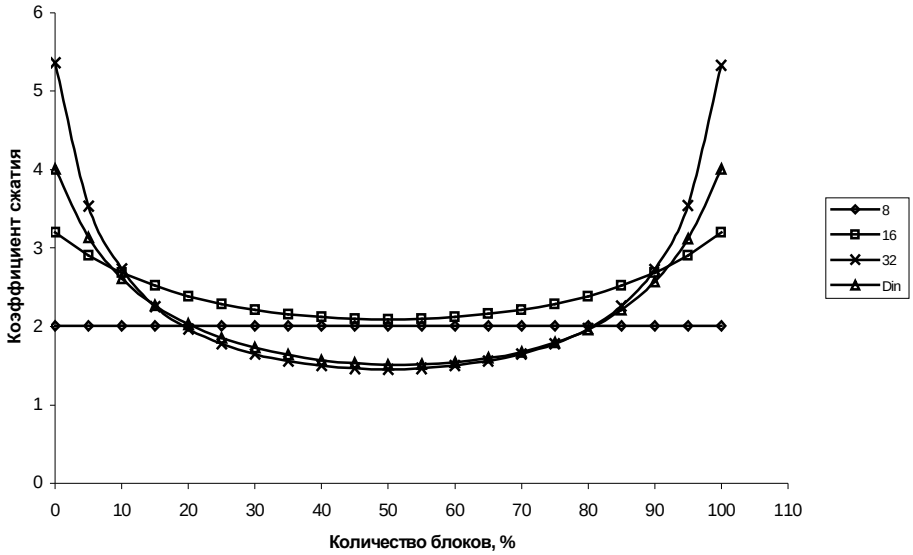

Рисунок 5 – Адресно-векторное сжатие данных с группами единиц по 8 в пакете (Din – динамическое сжатие)

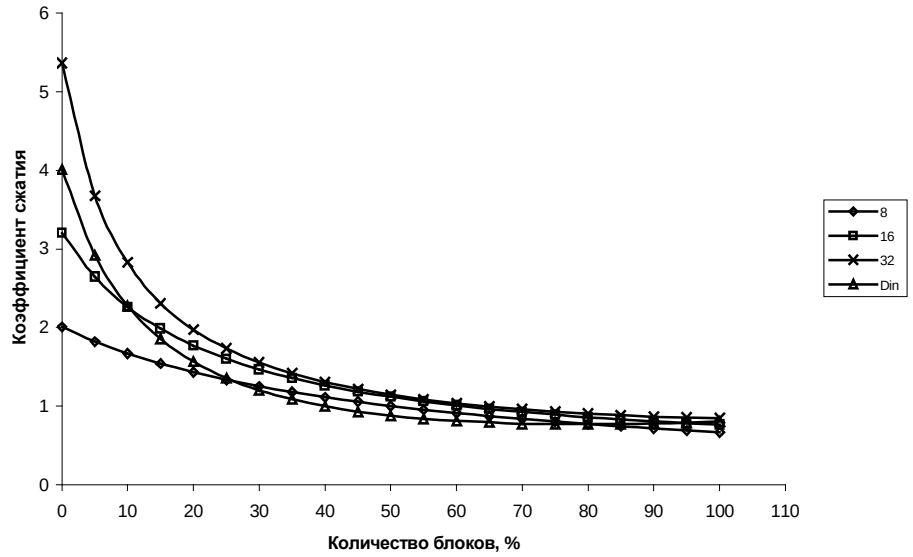

Рисунок 6 – Адресно-векторное сжатие данных с группами единиц по 4 в пакете (Din – динамическое сжатие)

Практическими результатами также подтверждается тот факт, что в случае преобладания двоичных единиц над нулями в кодовой комбинации сжатие можно произвести после инвертирования пакета. Поэтому графики на рисунке 5 симметричны относительно прямой, пересекающей ось абсцисс в точке 50% параллельно оси ординат.

При наихудшем с точки зрения адресно-векторного кодирования случае, когда количества единиц и нулей в обрабатываемом пакете равны (рисунок 6), алгоритмы статического адресно-векторного сжатия 8- и 16-разрядных комбинаций дают результат существенно хуже, чем динамический алгоритм (на определенном промежутке значений) даже с учетом соизмеримости заголовка пакета с размером содержащихся в нем данных, т.е. динамический подход к адресно-векторному сжатию дает выигрыш даже в этом случае.

ЗАКЛЮЧЕНИЕ

По результатам проведенных исследований и сравнительной оценки статических и динамического алгоритмов адресно-векторного сжатия можно сделать следующие выводы.

1 При рассматриваемых размерах обрабатываемых двоичных последовательностей, когда длина заголовка пакета соизмерима с длиной данных, эффективным является применение статического алгоритма адресно-векторного сжатия, так как он имеет минимальную длину служебной информации и прост в реализации.

2 С дальнейшим ростом размера пакета, когда служебная часть становится намного меньше длины самого пакета, динамическое адресно-векторное сжатие будет давать лучший результат по сравнению со статическим. Однако при этом возрастают аппаратурные затраты, связанные с необходимостью хранения больших объемов данных для обработки. Применение быстродействующих микроконтроллеров позволит обойти эту проблему и эффективно использовать динамические алгоритмы для быстрого адресно-векторного сжатия двоичных последовательностей.

SUMMARY

In order to get better coefficients of data address-vector compression more promising is the usage of dynamic algorithms that allow coders to adapt quickly to the current dataflow instead of the static ones.

СПИСОК ЛИТЕРАТУРЫ

1. Чернега В.С. Сжатие информации в компьютерных сетях: Учебное пособие для вузов Под ред. д-р тех. наук. Маригодова В.К. – Севастополь: СевГТУ, 1997. – 214 с.
2. Кричевский Р.Е. Сжатие и поиск информации. – М.: Радио и связь, 1989. – 168 с.
3. Lelewer D.A., Hirschberg D.S. Data compression // ACM Computing Surveys. – 1987. – Vol.9. – N. 3. – P. 261-296.
4. Кулик И.А. Об избыточности адресно-векторного кодирования // Вісник СумДУ. – 1996. – №1(5). – С. 90-93.
5. Кулик И.А. Оценка степени сжатия адресно-векторного кодирования // Вісник СумДУ. – 1996. – №2(6). – С. 84-87.
6. Кулик И.А. Синтез быстродействующих алгоритмов сжатия на основе адресно-векторного кодирования для информационных задач АСУ: Автореферат диссертации на соискание степени канд. техн. наук. – Харьков, 1998.

Поступила в редакцию 14 июня 2006 г.