Е.М. Кравцунов, Т.Р. Мустафин, д.т.н., проф. В.И. Перекатов
(ЗАО «МЦСТ», ОАО «ИНЭУМ им. И.С. Брука»)

E. Kravtsunov, T. Mustafin, V. Perekatov

КОММУНИКАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ СЕМЕЙСТВА «ЭЛЬБРУС» В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ УПРАВЛЕНИЯ: ПРОБЛЕМЫ СЖАТИЯ ДАННЫХ

COMMUNICATION OF «ELBRUS» COMPUTER MEANS IN DISTRIBUTED CONTROL SYSTEMS: DATA COMPRESSION PROBLEMS

В качестве теоретической основы процедур сжатия сигнального трафика в распределенных РМВ-системах управления рассматриваются побуквенное интервальное преобразование и кодирование Райса-Голомба с адаптацией к пересылаемому тексту. Практические заключения делаются по результатам модельных экспериментов с использованием тестовых наборов Calgary Corpus и зарегистрированной в ЗАО «МЦСТ» нагрузки по DNS-трафику.

Inversion frequencies transformation and Rice-Golomb coding with the adaptation to transmitted text are examined as the theoretical basis for data compression procedures in distributed RT-systems. Practical conclusions are drawn on the results of model experiments with the use of Calgary Corpus files and registered DNS-traffic load in the MCST local network.

Ключевые слова: сжатие данных, побуквенное интервальное преобразование, мощность алфавита, кодирование Райса-Голомба, информационная энтропия, теоретический предел сжатия, сжатие коротких блоков данных, аппаратная реализация.

Key words: data compression, inversion frequencies transformation, Rice-Golomb coding, alphabet power, information entropy, theoretical limit of compression, little data blocks compression, hardware implementation.

Введение

Результатом проектной деятельности ЗАО «МЦСТ», связанной с разработкой вычислительных средств для встраиваемых и мобильных применений, стало создание микропроцессоров и малогабаритных процессорных модулей на их основе, предназначенных для работы в реальном масштабе времени (РМВ). Реализованные в них архитектурные и конструктивно-технологические решения позволили выполнить требования к производительности вычислительного процесса при решении сложных задач управления удаленными объектами РМВ. Результаты этой работы достаточно подробно представлены в публикациях и Интернет-ресурсах компании. В то же время, расширение состава и функциональности системных объектов всё в большей степени ставит вопрос об оптимизации их взаимодействия, одним из методов которой является сжатие данных (compression), пересылаемых в системном трафике. В статье приводятся некоторые результаты исследований, проведенных в этом направлении.

Рассматривая намеченную сферу применения, авторы исходили из предположения о двух основных составляющих трафика: потоке сигналов от измерительных средств системы, например РЛС, и сообщений типа «запрос-ответ» в составе потока команд, пересылаемых между вычислительными модулями и исполнительными устройствами. В обоих случаях речь идет о сообщениях малой длины и близкого формата, что является хорошей предпосылкой для аппаратной реализации сжатия. С этой предпосылкой ставились следующие требования:

- однопроходная реализация сжатия или его основных стадий в режиме in-fly;
- несложный алгоритм поэлементной обработки трафика;
- минимальный объем памяти;
- выполнение сжатия данных с показателями, относительно близкими к теоретическому пределу.

Ввиду разнородности и, в определенных случаях, противоречивости этих установок, фактическая цель исследования состояла в том, чтобы оценить – в какой степени данное

сочетание можно осуществить на практике. В результате анализа возможных решений выбор был сделан в пользу схемы последовательной обработки потока данных, на первой стадии которой выполняется его побуквенное интервальное преобразование, на следующей – кодирование Райса-Голомба. Теоретические основы этих процессов изложены в разделе 1. Итоги текущей стадии исследования констатируются в Заключении на основании показателей модельных экспериментов, которые проведены на базе стандартных тестовых наборов и экспериментальных данных в разделе 2.

1. Однопроходные процедуры преобразования и кодирования данных со сжатием

1.1. Определения

Используются следующие понятия:

Алфавит $A[N]$ – множество символов, из которых состоит текст.

Мощность алфавита N – количеств символов в алфавите, $S(N)$ – количество битов, используемое для кодирования одного символа.

Теоретический предел сжатия K – количественная оценка максимального сжатия, при котором текст может быть восстановлен. Значение K в процентах определяется величиной $(1 - Q/S(N)) \times 100\%$. Здесь Q – информационная энтропия текста, равная минимальному количеству битов, которое необходимо для кодирования одного символа текста, имеющего определенную статистику. При использовании алгоритма побуквенных интервальных преобразований, рассчитанного на фиксированный алфавит, энтропия, согласно [1], задается формулой:

$$Q = \sum_{i=0}^{N-1} (q_i / D) \times \log_2 (q_i / D),$$

где q_i – количество символов i в тексте, D – длина текста.

Алфавитный порядок – порядок символов в алфавите, при котором $F(A[n]) > F(A[n + 1])$, где n принадлежит множеству $[0; N\text{-}1)$, $F(A[n])$ – частота появления символа $A[n]$ в тексте.

Лексикографически старший символ – из двух символов $A[i]$ и $A[j]$ алфавита A, в котором символы отсортированы в соответствии с алфавитным порядком, первый является лексикографически старшим, если $i > j$.

Смещение – количество символов, лексикографически старших текущего символа, которые встретились в тексте с момента его предыдущего появления или начала строки, если текущий символ встретился в тексте впервые.

1.2. Побуквенное интервальное преобразование текста с фиксированным алфавитным порядком символов

Принцип преобразования

Побуквенное интервальное преобразование текста с фиксированным алфавитным порядком символов[1] переводит набор символов текста в последовательность наборов смещения.

Рассмотрим исходную строку двухразрядных символов *ccdaaabbaaab* с алфавитным порядком {*a,b,c,d*}. Для символа *a* первое смещение 3 образуется символами *ccd.* Второе, третье, пятое и шестое смещения равны 0, т.к. на этих вхождениях символы *a* идут подряд. Четвертое смещение равно 2, т.к. между третьим и четвертым вхождениями символа *a* два раза встречается лексикографически старший символ *b.* Соответственно, для символа *a* последовательность смещений имеет вид {3;0;0;2;0;0}. Вследствие того, что вхождения младших символов не влияют на смещения старших, далее можно рассматривать исходную строку без символа *a*: *ccdbbb.* Относительно этой строки набор смещений для символа *b* образует последовательность{3;0;0}, для символа *c* – {0;0}, для *d* – {0}. Если подсчитать количество элементов в каждом из полученных наборов смещений и расположить эти значения в порядке от *a* до *d*, то получим последовательность {6;3;2;1}.

Таким образом, побуквенное интервальное преобразование позволило преобразовать

[1] В заглавии раздела использовано общепринятое название преобразования, которое по существу рассматривается применительно к объектам текста, обозначаемым здесь как «символы».

реальное представление исходной строки в структуру, которая однозначно характеризует количество и порядок вхождения каждой буквы в исходный текст (табл. 1). В общем случае такая структура представлена в табл. 2, где $d(i)$ – количество смещений для i-й буквы, $n(i)(j)$ – j-е смещение для i-й буквы, $j = 0 \ldots d(i)-1$.

Таблица 1

Представление исходной строки в результате побуквенного интервального преобразования

Номер буквы	**Количество смещений**	**Последовательность смещений**
0	6	3 0 0 2 0 0
1	3	3 0 0
2	2	0 0
3	1	0

Таблица 2

Обобщенное представление строки в результате побуквенного интервального преобразования

0	$d(0)$	$n(0)(0)$; $n(0)(1)$; …; $n(0)(j)$; …; $n(0)(d(0)-2)$; $n(0)(d(0)-1)$
1	$d(1)$	$n(1)(0)$; $n(1)(1)$; …; $n(1)(j)$; …; $n(1)(d(1)-2)$; $n(1)(d(1)-1)$
...	...	...
i	$d(i)$	$n(i)(0)$; $n(i)(1)$; …; $n(i)(j)$; …; $n(i)(d(1)-2)$; $n(i)(d(1)-1)$
...	...	...
$N-2$	$d(N-2)$	$n(N-2)(0)$; $n(N-2)(1)$; …; $n(N-2)(j)$; …; $n(N-2)(d(N-2)-2)$; $n(N-2)(d(N-2)-1)$
$N-1$	$d(N-1)$	$n(N-1)(0)$; $n(N-1)(1)$; …; $n(N-1)(j)$; …; $n(N-1)(d(N-1)-2)$; $n(N-1)(d(N-1)-1)$

Однопроходная реализация в кодере передатчика

В [2] описывается процедура, основанная на использовании бинарного дерева при обработке очередного блока символов. На рис. 1 в качестве примера взят блок, который соответствует исходной последовательности символов, приведенной выше. Листья дерева помечены буквами алфавита так, что более левым листьям соответствуют лексикографически меньшие символы.

Выше было принято, что определенный символ имеет индекс i среди всех выстроенных по старшинству символов алфавита. Параметрами процедуры преобразования являются:

– счетчики в листьях дерева;

– счетчики в узлах более высоких уровней;

– переменная G, которая при обработке очередного символа получает значение, равное смещению от начала блока;

– переменные G_i, каждая из которых равна значению G, полученному при предыдущем вхождении данного символа;

– $Offset[i]$ – элемент вектора *Offset*, в котором накапливаются последовательности смещений данного символа.

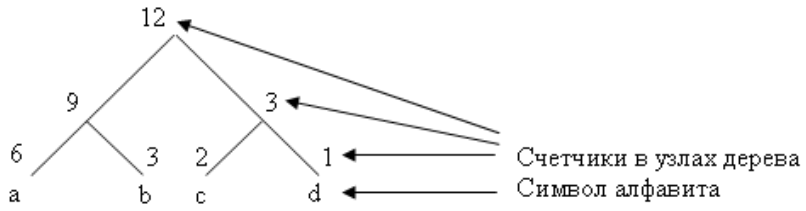

Рис. 1

Бинарное дерево после обработки блока *ccdaaabbaaab*

Перед обработкой блока все параметры инициализируются нулем. При выборке очередного символа из блока, выполняются следующие операции:

– к счетчику в соответствующем листе дерева прибавляется 1. Таким образом, на этом уровне накапливается количество предыдущих вхождений для всех символов алфавита;

– совершается проход по цепи узлов более высокого уровня от листа к корню, на каждом шаге которого к счетчику узла прибавляется 1, после чего значение счетчика правого братского узла прибавляется к G. В силу того что счетчики в правых братских узлах содержат значения для лексикографически старших символов, после обработки корня все предыдущие старшие символы в G учтены;

– разность $G - G_i$, равная очередному смещению данного символа, записывается в элемент $Offset[i]$.

В результате после обработки блока процедурой преобразования сформировано всё содержание табл. 2: в счетчиках листьев заданы количества смещений, в векторе *Offset* – последовательности смещений. Алгоритм обратного побуквенного преобразования, построенный на сходной основе, приведен в [2].

Реализация сжатия применительно к алгоритмам на основе бинарного дерева возможна при введении динамической настройки, которая ассоциирует листы дерева с определенными символами соответственно частоте их вхождения в предшествующий блок текста.

1.3. Кодирование Райса-Голомба

Принцип кодирования

Кодируемое двоичное число разбивается на две части: k младших разрядов и все остальные разряды от k-го и старше. Если значение числа, представляемого (в двоичном виде) в старшей части равно V, то старшая часть заменяется V единицами, за ней следует разделительный нуль, после этого записывается младшая часть в неизменном виде.

Рассмотрим число $38 = 00100110_2$ (8 битов). Его код с параметром $k = 4$ формируется следующим образом:

$0010\ 0110_2$ → старшая часть $0010_2 = 2$ представляется как 11 (две единицы) + разделительный 0 + младшие 4 бита 0110_2 без изменения → итоговый код: 1100110 (7 битов).

Используем этот принцип для представления первой строки табл. 1 кодом с параметром $k = 1$.

Количество смещений:

$6 = 110_2$ → три единицы + разделительный ноль + один бит младшей части 0_2 → 11100_2.

Последовательность смещений:

$3 = 11_2$ → единица + разделительный ноль + один бит младшей части 1_2 → 101_2,

$0 = 0_2$ → ноль единиц + разделительный ноль + один бит младшей части 0_2 → 00_2

и так далее – до конца первой строки и по всем строкам.

В результате этой процедуры получим представление исходной строки символов в виде (пробелами для наглядности разделены закодированные строки табл. 1):

1110010100001000000 1011010000 1000000 0100.

По сути те же действия выполняются и в общем случае, представленном в табл. 2: при последовательном проходе алгоритма кодирования с параметром *k* по ее строкам для каждой строки *i* сначала формируется код количества смещений *d(i)*, а за ним (в порядке записи) коды элементов последовательности смещений:

$$n(i)(0);\ n(i)(1);\ \ldots;\ n(i)(j);\ \ldots;\ n(i)(d(1)\text{-}2\ );\ n(i)(d(1)\text{-}1).$$

Адаптация параметра кодирования к составу текста

Побуквенное интервальное преобразование с использованием кодирования Райса-Голомба в общем случае не обеспечивает сжатие данных: взятая в качестве примера символьная строка содержит 24 бита, в то время как после кодирования ее длина увеличивается до 40 битов.

В качестве ключевого фактора, определяющего эффективность сжатия при кодировании Райса-Голомба, в данном исследовании было взято значение параметра *k*. Малое значение *k* при большом значении старшей части сжимаемого числа может привести к резкому увеличению количества представляющих ее единиц, с другой стороны, при слишком большом значении *k* в представление младшей части будут входить избыточные нули, если ее значение невелико. Степень влияния этих негативных эффектов на общий результат сжатия зависит от конкретного вида последовательности смещений. По этой причине реальную перспективу применения имеет следующий метод динамической адаптации *k* в процессе сжатия, разработанный на основе положений [3] и описанный в [4].

В результате побуквенного интервального преобразования подлежащие сжатию данные приводятся в передатчике к форме (известной и в приемнике), представленной строками табл. 2. Строки, начиная с нулевой, последовательно подаются на вход коди-

ровщика Райса-Голомба, способного динамически адаптировать значения k в пределах от 0 до 20 к содержанию строки. С этой целью используется вспомогательный вектор *Sum* с 21 элементом.

Для каждой строки i циклическая процедура кодирования выглядит следующим образом.

Инициализация:

1) Массив *Sum* инициализируется нулями, параметр k – значением k_{i0}.

2) Кодируется количество смещений $d(i)$.

3) В качестве текущего смещения берется нулевое смещение ($j = 0$).

Тело цикла:

1) Смещение номер j кодируется текущим значением k.

2) Для каждого $k = 0 \ldots 20$ рассчитывается длина кода смещения $n(i)(j)$ и прибавляется к соответствующему элементу массива *Sum*.

3) В массиве *Sum* определяется индекс минимального элемента, текущее значение k меняется на найденный индекс.

4) Следующее смещение становится текущим: $j = j + 1$. Переход к пункту 1.

В процессе обработки смещений в каждой строке табл. 2 очередной выбор параметра k основан на результатах обработки каждого из предыдущих смещений, заданных в данной строке. Это позволяет однозначно выполнить восстановление строки, если декодеру известно значение k_{i0}.

В [4] для определения k_{i0} предложена эмпирическая формула

$$k_{i0} = 1 + [\log_2(N - i + 1)],\ i = 0 \ldots (N - 1),$$

результативность которой подтверждена экспериментальным путем.

В декодере приемника это значение позволяет восстановить закодированное нулевое смещение, поступившее от передатчика. Найдя с его использованием значение k по тому же принципу, что был при передаче, можно восстановить первое смещение и так далее до

конца строки.

2. Ключевые аспекты реализации

Моделирование на стандартном тестовом наборе

В качестве возможных альтернатив рассматривались алфавиты с разрядностью символов 2 (4 символа), 4 (16 символов), 8 (256 символов). Измерения проводились на тестовом наборе Calgary Corpus, включающем 18 файлов с несколькими типами форматов. Значения коэффициентов сжатия, приведенные в табл. 3, усреднены относительно показателей, полученных для каждого файла.

Таблица 3

Усредненные коэффициенты сжатия для тестового набора Calgary Corpus

	Экспериментальный коэффициент сжатия			Теоретический предел коэффициента сжатия		
Мощность алфавита	4	16	256	4	16	256
Среднее	1,26	15,31	36,50	7,63	17,61	38,86

Моделирование показало, что экспериментальный результат для алфавита с минимальной разрядностью далек от теоретического предела, в то время как для алфавитов с 16-ю и 256-ю символами можно констатировать достаточную близость теоретического предела и экспериментально полученного коэффициента сжатия. В этой связи уместно отметить, что, несмотря на предпочтительную эффективность сжатия для алфавита из 256 символов, этот выбор предполагает аппаратную реализацию сложного бинарного дерева, имеющего 8 уровней, 256 листьев и 512 узлов.

Исследования на сетевом трафике типа DNS

При постановке этих исследований учитывалось, что трафик сообщений между объектами систем управления РМВ можно поставить в соответствие трафику сообщений протокола DNS в сети Интернет, передаваемых короткими дейтаграммами UDP.

Зарегистрированная в сети ЗАО «МЦСТ» часть общего трафика DNS была представлена маленькими тестовыми пакетами с длиной поля данных в пакете UDP от 12 до

500 байтов. Результаты ее обработки с использованием алфавитов из 16 и 256 символов представлены в табл. 4.

Таблица 4

Результаты эксперимента по сжатию трафика пакетов DNS

Размер алфавита, символы	**Размер сегмента данных, байт**	**Всего пакетов**	**Среднее сжатие, %**
16	До 300	17025	11,5
	От 300 до 500	830	13,4
256	До 300	17025	-29,7
	От 300 до 500	830	-18,4

Наиболее значимый из этих результатов заключается в том, что отмеченная для трафика Calgary Corpus эффективность алфавита из 256 символов в данном случае совершенно теряется, более того, наблюдается обратный эффект (decompression). Это вызвано большими издержками процедуры побуквенного преобразования, в которой надо задавать количество смещений для каждого из 256 символов применительно к небольшим сообщениям.

Заключение

На основании приведенных результатов наиболее (а может быть, единственно) приемлемым вариантом сжатия данных, который бы в общем удовлетворял совокупности приведенных во введении требований, является использование побуквенного интервального преобразования и кодирования Райса-Голомба на основе алфавита из 16 символов. В распределенных системах управления с мобильными и встраиваемыми вычислительными средствами ЗАО «МЦСТ», применительно к которым и проводилось исследование, это реально выполнить путем применения несложных аппаратных модулей, обеспечивающих средние показатели сжатия в районе 12%. При расчете на небольшой размер пакетов целесообразность этого решения должна определяться в зависимости от специфики конкретной системы. Любое расширение взятой на данный момент сферы применения предпола-

гает дальнейшие исследования.

Литература

1. Shannon C. A mathematical theory of communication – Bell System Technical Journal, 1948, v. 27, p. 379-423 and p. 623-656.

2. Кадач А.В. Сжатие текстов и гипертекстов – «Программирование», 1997, № 4, с. 47-56.

3. Браиловский И.В. Эффективное сжатие картографических изображений без потерь качества. – Тр. 6-й международной конференции «Распознавание образов и анализ изображений», В. Новгород, 2002, с. 85-87.

4. Кравцунов Е.М. Эффективные методы сжатия данных для встраиваемых систем – Сб. науч. тр. ИМВС РАН «Высокопроизводительные вычислительные системы и микропроцессоры», вып. 6. – М., 2004.