САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Системного программирования

Быкова Юлия Сергеевна

Поиск оптимального решения задачи расчета синдромов в RAID N+M

Курсовая работа

Научный руководитель:
Разработчик исследовательской лаборатории RAIDIX Маров А. А.

Санкт-Петербург
2014

Оглавление

Введение

Системы хранения данных (СХД) - это комплексное решение для хранения больших объемов информации, а так же быстрого и бесперебойного доступа к ней. Одновременное использование нескольких жестких дисков в СХД позволяет увеличить объем хранимой информации и распараллелить чтение и запись данных, ускоряя, тем самым, процесс работы с ними.

Из множества различных технологий СХД в данной работе будет рассматриваться RAID (Redundant Array of Independent Disks) - «отказоустойчивый массив из независимых дисков». Концепция структуры состоит из нескольких дисков, объединенных в группу. Для обеспечения отказоустойчивости массивов на них записываются и хранятся не только данные, но и некоторая избыточная информация (синдромы), которая позволяет восстанавливать данные в случае их частичной утраты. RAID N+M означает, что в массиве на N дисков с данными будет использоваться M дисков с синдромами.

В технологии выделяется несколько проблем: расчет синдромов, восстановление дисков и SDC (Silent data corruption) - исправление скрытых повреждений. В данной работе внимание уделяется первой из них.

Цели

Нахождение оптимального решения задачи расчета синдромов в поле $GF(2^8)$.

Проведение исследования, анализ и реализация алгоритмов.

Задачи

Найти, проанализировать и модернизировать алгоритмы расчета контрольных сумм.

Написать рабочий прототип модуля.

Векторизовать вычисления.

Провести замеры производительности.

1. Поле Галуа

Для расчета контрольных сумм часто используется арифметика конечных полей Галуа $GF(2^8)$ [5]. Работа происходит не с отдельными битами, а с векторами, которыми являются полиномы, как элементы конечного поля. Для элементов поля определены основные операции, такие как сложение, умножение и деление.

Поле Галуа $GF(p^q)$ - это поле из p^q элементов. $GF(p^q) = \mathbb{Z}_p[x]/\langle f(x)\rangle$, где $f(x)$ - неприводимый многочлен степени q над $\mathbb{Z}_p$, p - простое число. Элементами поля $GF(p^q)$ являются многочлены степени меньшей q с коэффициентами из $\mathbb{Z}_p$. Сложение $a(x)$ и $b(x)$ из поля Галуа получается, как сложение многочленов с приведением коэффициентов по модулю p [4]. Операция умножения $a(x) * b(x)$ определяется как произведение многочленов по модулю $f(x)$ с приведением коэффициентов по модулю p.

При кодировании рассматривают чаще всего $p = 2$, так как информация представляется в виде множества из {0,1} [2]. Прошлогодние исследования показали, что оптимальная степень поля на данный момент $q = 8$. При кодировании информации с использованием арифметики конечных полей Галуа $GF(2^8)$ каждому вектору сопоставляется полином степени меньше 8.

2. Коды Рида-Соломона

Коды Рида-Соломона относятся к блочным, помехоустойчивым кодам и могут использоваться в области хранения информации для избегания потери поврежденной информации. Они являются частным случаем БХЧ кодов, представляют собой циклические коды, позволяющие исправлять ошибки в блоках данных [7].

Порождающий полином кода Рида-Соломона - полином минимальной степени над полем $GF(2^q)$, корнями которого являются подряд идущие степени примитивного элемента поля.

Пусть $g(x) = (x + a^{l_0})(x + a^{l_0+1})...(x + a^{l_0+2t-1})$ - порождающий полином кода, здесь a – примитивный элемент поля $GF(2^q)$, l_0 - некоторое целое число, если взять $l_0 = 0$, то порождающий полином кода примет вид $g(x) = (x + 1)(x + a)...(x + a^{2t-1})$.

Пусть имеются кодовые слова $(D_0, D_1, ..., D_{n-1}), D_i \in GF(2^q)$. Рассмотрим полином степени n над $GF(2^q)$ с коэффициентами равными $(D_0, D_1, ..., D_{n-1})$.

$G(x) = D_0x^{n-1} + D_1x^{n-2} + ... + D_{n-2}x + D_{n-1}$. Для кода, исправляющего t ошибок с неизвестными номерами, необходимо как минимум $2t$ избыточных кодовых слов. Тогда введем полином $G^*(x) = G(x) * x^{2t} = D_0x^{n-1+2t} + D_1x^{n-2+2t} + ... + D_{n-2}x^{2t+1} + D_{n-1}x^{2t}$.

Для вычисления синдромов разделим полином $G^*(x)$ на $g(x)$ и получим остаток $R(x)$. Имеем $G^*(x) = Q(x)g(x) + R(x)$. Здесь $deg(G^*(x)) = 2t + n - 1, deg(Q(x)) = n - 1, deg(g(x)) = 2t, deg(R(x)) = 2t - 1$. В качестве синдромов берутся коэффициенты полинома $R(x) = C_0x^{2t-1} + C_1x^{2t-2} + ... + C_{2t-1}$.

Итоговый кодовый полином имеет вид $\bar{G}(x) = G^*(x) + R(x)$. Для этого полинома выполняется $\bar{G}(1) = 0, \bar{G}(a) = 0, ..., \bar{G}(a^{2t-1}) = 0$. По каналу связи передается сообщение вида $(D_0, D_1, ..., D_{n-1}, C_0, ..., C_{2t-1})$, т.е. полином $\bar{G}(x)$. Если при передаче происходит повреждение некоторых кодовых слов переданного сообщения, тогда, посчитав значение полинома $\bar{G}(x)$ в корнях полинома образующего код, получим $\bar{G}(1) = \tilde{S}_0, \bar{G}(a) = \tilde{S}_1, ..., \bar{G}(a^{2t-1}) = \tilde{S}_{2t-1}$. Полученные значения $\tilde{S}_i, i = 0..2t - 1$ используются для построения полинома локаторов ошибок. Найдя корни этого полинома, можно определить исходные значения поврежденных блоков.

3. Алгоритмы расчета синдромов

В RAID массивах диски разбиваются на блоки одинакового размера. Последовательность блоков с одинаковыми номерами, расположенных на разных дисках, называют страйпом. Расчет контрольных сумм и восстановление данных производится по страйпам, поэтому в дальнейшем, говоря о дисках, будем понимать различные блоки одного страйпа. И все вычисления производить с блоками страйпа. [1] [3]

Пусть у нас есть информацонные блоки $D_0, D_1, ..., D_{n-1} \in GF(2^8)$. Посчитаем синдромы $S_0, S_1, ..., S_{m-1} \in GF(2^8)$ такие, что контрольные суммы, посчитанные от всей последовательности блоков, нулевые.

Матричный метод

Опр: Матрица Вандермонда

$$V_{m,n}(\beta_1, \beta_2, ..., \beta_n) = \begin{pmatrix} \beta_1^0 & \beta_2^0 & \dots & \beta_n^0 \\ \beta_1^1 & \beta_2^1 & \vdots & \beta_n^1 \\ \vdots & \vdots & \dots & \vdots \\ \beta_1^{m-1} & \beta_2^{m-1} & \dots & \beta_n^{m-1} \end{pmatrix}$$

Тогда задача может быть записана следующим образом: найти такие $S_0, S_1, ..., S_{m-1}$, что

$$V_{m,n+m}(a^{n+m-1}, a^{n+m-2}, ..., a, 1) \begin{pmatrix} D_{n,1} \\ S_{m,1} \end{pmatrix} = \mathbb{O}_{m,1}$$

Здесь $D_{n,1} = (D_0, D_1, ..., D_{n-1})^T, S_{m,1} = (S_0, S_1, ..., S_{m-1})^T$,$\mathbb{O}_{m,1}$ – нулевой m-мерный вектор столбец.a – примитивный элемент поля.

После нескольких преобразований с помощью свойств матриц и полей уравнение примет вид

$$S_{m,1} = G_{m,n} D_{n,1},$$

где $G_{m,n}$ - некая матрица, не зависящая от данных. Данный факт очень важен, так как теперь можно вычислить матрицу заранее.

Полиномиальный метод

Рассмотрим другой подход к расчету синдромов.

Обозначения:

Образующий многочлен кода: $g(x) = (x + a^{l_0})(x + a^{l_0+1})...(x + a^{l_0+2t-1})$, $degg(x) = m$

Информационный многочлен: $G(x) = D_0 x^{n-1} + D_1 x^{n-2} + ... + D_{n-2} x + D_{n-1}$, $degG(x) = n - 1$ - многочлен над $GF(2^k)$

1. Умножим информационный многочлен на x^m

   $\bar{G}(x) = G(x)x^m = (D_0 x^{n-1} + D_1 x^{n-2} + ... + D_{n-1})x^m$, $deg\bar{G}(x) = n + m - 1$

2. Найдем остаток от деления $\bar{G}(x)$ на g:

   $\bar{G}(x) = Q(x)g(x) + R(x)$,

   $degQ = n-1, degR = m-1$

3. Рассмотрим $G^*(x) = \bar{G}(x) + R(x)$, для него выполняется

   $G^*(1) = 0$

   $G^*(a) = 0$

   ...

   $G^*(a^{m-1}) = 0$

   - Синдромы от всей последовательности блоков.

   Следовательно, искомые $S_0, S_1, ..., S_{m-1}$– коэффициенты $R(x)$

Как найти эти коэффициенты?

$g(x)$ представляет собой произведение линейных многочленов. Тогда рассмотрим нахождение остатка при делении $\bar{G}(x)$, как последовательное деление на эти линейные многочлены по схеме Хорнера

Таблица 1: Схема Хорнера

	D_0	D_1	...	D_{n-1}	0	0	...	0
1	D_0	$D_0 + D_1$	...	...	...	...	...	r_1
a	...	...	...	...	...	...	r_2	
a^2	...	...	...	...	...	r_3		
...	...	...	...	...	...			

При последовательном делении на линейные многочлены получали:
$\bar{G}(x) = (x+1)q_1 + r_1$, далее $q_1 = (x+a)q_2 + r_2$, $q_2 = (x+a^2)q_3 + r_3$ и т.д.
Тогда подставив эти соотношения, получим:
$\bar{G}(x) = (x+1)[(x+a)q_2 + r_2] + r_1 =$
$= (x+1)(x+a)q_2 + (x+1)r_2 + r_1 =$
$= (x+1)(x+a)(x+a^2)q_3 + (x+1)(x+a)r_3 + (x+1)r_2 + r_1$
и т.д.
Подставив все, что нашли по схеме Хорнера в представление $\bar{G}(x)$ получится
$\bar{G}(x) = (x+1)(x+a)(x+a^2)...(x+a^{m-1})q_m + R_{m-1}(x) = Q(x)g(x) + R_{m-1}(x)$
Коэффициенты $R_{m-1}(x)$ нас и интересуют
Но
$R_{m-1}(x) = (x+1)(x+a)(x+a^2)...(x+a^{m-2})r_m +$
$+(x+1)(x+a)(x+a^2)...(x+a^{m-3})r_{m-1} + ... +$

$+(x+1)(x+a)r_3 + (x+1)r_2 + r_1$

Введем обозначения:

$f_{m-1}(x) = (x+1)(x+a)(x+a^2)...(x+a^{m-2}), degf_m - 1 = m-1$

$f_{m-2}(x) = (x+1)(x+a)(x+a^2)...(x+a^{m-3}), degf_m - 2 = m-2$

...

$f_1(x) = (x+1), f_0(x) = 1$

С учетом обозначений

$R_{m-1}(x) = f_{m-1}(x)r_m + f_{m-2}(x)r_{m-1} + ... + f_1(x)r_2 + f_0(x)r_1$

Рассмотрим один из введенных полиномов, например, $f_{m-1}(x)$ – для него $1, a, ..., a^{m-2}$ корни. Этот полином можно записать в виде

$f_{m-1}(x) = x^{m-1} + f_{m-1}^{(1)}x^{m-2} + f_{m-1}^{(2)}x^{m-3} + ... + f_{m-1}^{(m-1)}$

Где $f_i^{(j)}$ – коэффициенты многочленов. Нижний индекс показывает многочлену какой степени принадлежит коэффициент. Верхний индекс – какой по счету этот коэффициент. Так как мы знаем корни многочлена, то по формулам Виета:

$f_{m-1}^{(1)}(x) = 1 + a + a^2 + ... + a^{m-2}$

$f_{m-1}^{(2)}(x) = 1 * a + 1 * a^2 + ... + 1 * a^{m-2} + aa^2 + ... + a^{m-3}a^{m-2}$

$f_{m-1}^{(3)}(x) = 1 * a * a^2 + 1 * a * a^3 + ... + a^{m-4}a^{m-3}a^{m-2}$

...

$f_{m-1}^{(m-1)}(x) = 1 * a * a^2 * ... * a^{m-2}$

Поэтому можем легко найти все коэффициенты этих полиномов. При вычислении стоит помнить, что $a^i * a^j = a^{(i+j)mod(2^k-1)}$. Зная все коэффициенты полиномов, помня, что

$R_{m-1}(x) = f_{m-1}(x)r_m + f_{m-2}(x)r_{m-1} + ... + f_1(x)r_2 + r_1$

и собрав полученные коэффициенты при соответствующих степенях x вектор коэффициентов $R_{m-1}(x)$ запишем следующим образом

$$\begin{pmatrix} r_m \\ r_{m-1} + f_{m-1}^{(1)}r_m \\ r_{m-2} + f_{m-2}^{(1)}r_{m-1} + f_{m-1}^{(2)}r_m \\ ... \\ r_1 + f_1^{(1)}r_2 + ... + f_{m-2}^{(m-2)}r_{m-1} + f_{m-1}^{(m-1)}r_m \end{pmatrix} = \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ ... \\ S_{m-1} \end{pmatrix}$$

4. Сравнение алгоритмов

У каждого метода есть как плюсы, так и минусы.

Матричный метод:

1. Матрицу G можно вычислить заранее
2. Синдромы вычисляются параллельно
3. Требуется дополнительное место для хранения матриц G

Полиномиальный метод:

1. Для расчета r_i по схеме Хорнера используются только умножения на степени примитивного элемента и сложения элементов поля.
2. Коэффициенты $f_i^{(j)}$ можно вычислить заранее.
3. Для вычисления коэффициентов r_i по схеме Хорнера требуется дополнительная память.

Ниже приведена таблица сравнения количества операций сложения и умножения различных алгоритмов.

Таблица 2: Количество операций

Способ расчета	Количество операций	
	умножения	сложения
Матричный	mn	$m(n-1)$
Полиномиальный	$mn + \frac{m(m-1)}{2}$	$mn + \frac{m(m-1)}{2}$

Как видно из таблицы, матричный метод использует значительно меньше операций как умножения, так и сложения. Несмотря на это еще возможен выигрыш полиномиального метода. В матричном методе происходит перемножение двух чисел размера 16 байт, в то время как в полиномиальном 16 - ти байтного числа на 1 байтное.

5. Инструменты

Был применён кодогенератор. Для каждого возможного количества дисков в массиве (от 5 до 128) и для каждого возможного количества избыточных дисков (от 2 до 64) по задумке должна быть своя функция расчета синдромов. Это позволяет реализовать их наиболее оптимальным образом для каждого конкретного случая, избегая использования условных переходов и бесполезных участков кода. Для того, чтобы не писать однотипные функции вручную, был реализован их генератор, что также позволило вносить изменения, направленные на оптимизацию, во все функции сразу.

Код был написан на языке Си. Язык Си был выбран по следующим причинам:

1. Си является языком верхнего уровня. Написание и отладка кода на нем производится значительно быстрее, чем на ассемблере.

2. Исходный код, написанный на Си, может быть перенесен в системы с процессорами различных архитектур.

3. Оптимизации компилятора языка Си позволяют компилировать высокопроизводительный код. Именно высокопроизводительные вычисления необходимы в СХД для высокой скорости доступа к данным. [6] Так был использован компилятор gcc 4.7, и код собирался с уровнем оптимизации –O2.

Все расчеты проводились с помощью технологии SSE, которая позволила векторизовать наши алгоритмы.

6. Реализация и тестирование

Используя приведенные выше методы, был реализован расчетный модуль для работы RAID с M синдромами. Расчетный модуль выполняет операции по расчету контрольных сумм с помощью двух различных алгоритмов.

Для всех возможных конфигураций систем было проведено тестирование корректрости работы алгоритмов. Тестирование осуществлялось за счет теста коррекотности, которому при запуске передавались параметры: количество жестких дисков в СХД и длина страйпа СХД.

Простейший тест корректности включает в себя следующие шаги:

1. Для выбранных размера страйпа S и количества дисков D заполняется случайными данными выделенная область памяти M соответствующего размера (S*D), логически разделяемая на D подряд идущих блоков, размера S, эмулирующие собой различные жесткие диски СХД.

2. Запускается функция расчёта контрольных сумм, которые сохраняются в служебных блоках (это последние блоки в дисковом массиве).

Замеры проводились следующим образом: до и после каждого вызова тестируемой функции вызывается команда, возвращающая значение счётчика процессорных тиков; использование этой команды позволило добиться высокой точности измерений. Для каждой функции тест проводился 1000 раз, затем обрасывались по 5% самых маленьких и самых больших результатов. Из оставшихся берётся среднее значение.

Используемая конфигурация тестовой машины:

ОС: Debian 7.0

Тип ОС: x64

CPU: Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz 2.40GHz

Оперативная память: 7,89 Гб

7. Результаты измерений

Ниже приведены графики сравнения различных алгоритмов расчета синдромов.

На всех графиках по оси X отложено количество дисков, а по оси Y – скорость расчета (в Мб) одного вызова функции расчета в тысячах тиков процессора

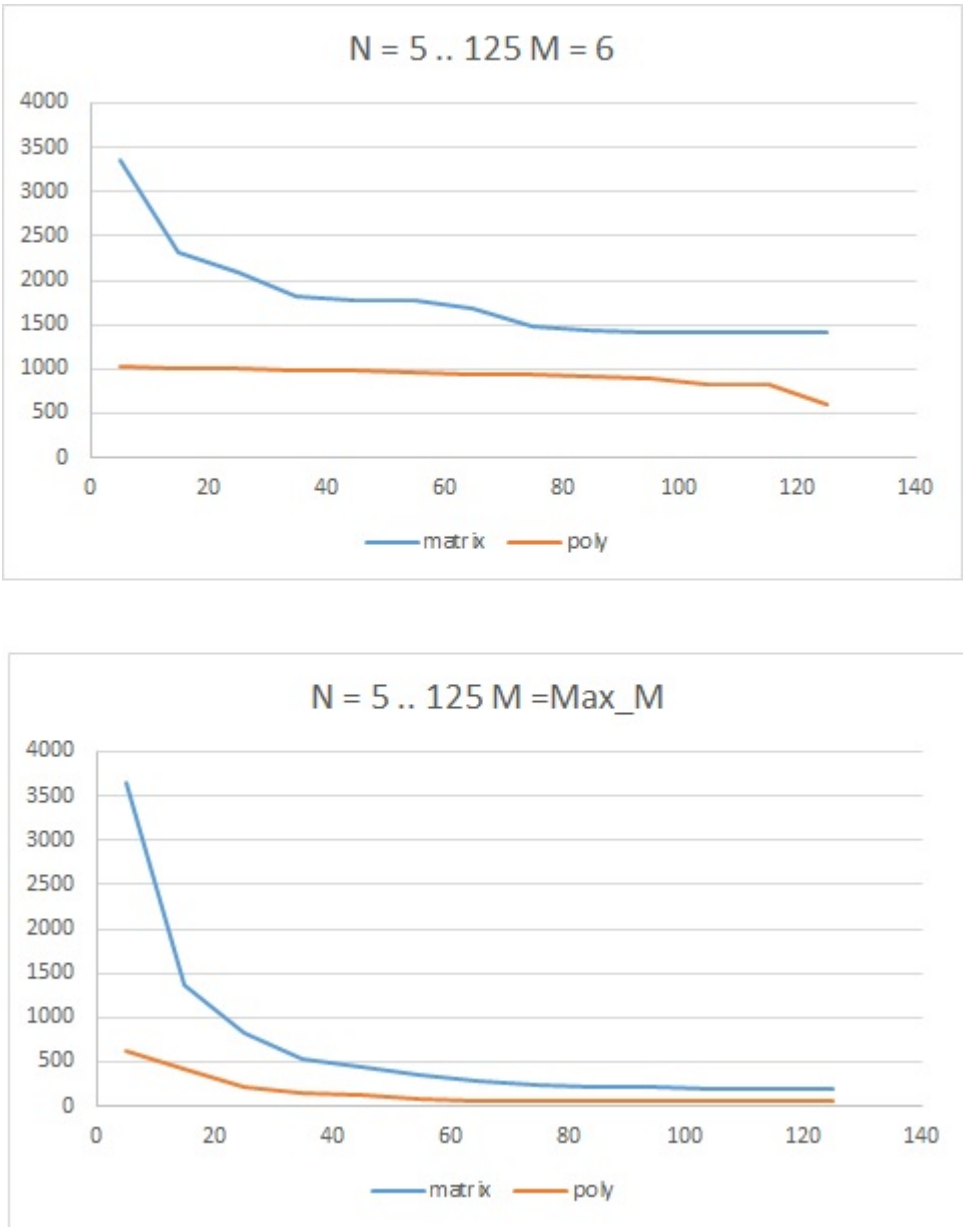

Рис. 1: График сравнения

Matrix – матричный метод, poly – полиномиальный метод

$Max_M = max(N, 64)$ - максимальное количество синдромов для каждого N

Из графика видно, что полиномиальный метод существенно уступает в производительности матричному методу. К сожалению, даже упрощенное умножение не сыграло особой роли. Стоит заметить, что при малых M и N отношение скорости двух методов значительно выше, чем при максимальных. На малых отношение примерно

в 7 раз, в то время как при больших размерах всего в 2 раза. Возможно, если бы мы увеличили количество дисков, то пришли бы к точке, в которой полиномиальный метод выигрывает, но в рамках данной работы это не рассматривается.

Заключение

В данной работе рассматривался подход к вычислениям в RAID с использованием арифметики конечных полей GF(28). Разработаны код-генераторы для генерации кода функций расчета синдромов. Подготовлена среда испытаний, программа тестов производительности. Проведены тесты корректности и замеры производительности получившихся функций.

Выполнен анализ полученных результатов. Сделано и проверено предположение о нахождении оптимального алгоритмы расчета, выявлены некоторые причины этого.

В дальнейшем возможна реализация с помощью технологии AVX. Также можно применять различные модификации к матричному алгоритму.

Список литературы

[1] Edge Mann S. Anderson M. Rychlik M. On the Reliability of RAID Systems: An Argument for More Check Drives. — http://arxiv.org/pdf/1202.4423v1.pdf.

[2] H.P. Anvin. Mathematics of RAID-6. — http://kernel.org/pub/linux/kernel/people/hpa/raid6.p

[3] S. Plank J. A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems. — http://web.eecs.utk.edu/ plank/plank/papers/CS-96-332.pdf.

[4] А.Ю Утешев. Поля Галуа. — http://pmpu.ru/vf4/gruppe/galois.

[5] Г. Лидл Р. Нидеррайтер. Конечные поля в 2-х т. Т. 1. — М. : Мир, 1988.

[6] Д.Э. Кнут. Искусство программирования. — Т.2:Полученные алгоритмы. : Издательство Вильямс, 2004.

[7] Э. Берлекамп. Алгебраическая теория кодирования. — М. : Мир, 1971.