САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
Математико-механический факультет
Кафедра системного программирования

Реализация алгоритмов расчета RAID 6 с использованием встроенных функций SSE

Курсовая работа студента 345 группы
Макулова Рустама Наилевича

Научный руководитель Короткевич А.И.

Санкт-Петербург
2012

СОДЕРЖАНИЕ

Обозначения и сокращения

СХД	Система хранения данных
RAID	Redundant Array of Independent Disks

ВВЕДЕНИЕ

В современном мире наблюдается быстрый рост хранимой информации. Растет не только объем хранимых данных, но и требования к обеспечению сохранности, целостности и восстановления после сбоев, случающихся в устройствах хранения.

Во многих случаях это достигается путем использования избыточного массива независимых жестких дисков (RAID). Существует несколько уровней таких массивов (0, 1, 2 и пр.) и каждый из них предназначен для выполнения различных функций. Вместе с этим становится актуальной задача повышения производительности RAID.

Исследованием скорости работы и выявлением оптимальной реализации занималась наша группа, в рамках работы которой выполнялась данная курсовая работа.

Целью данной работы было изучение алгоритмов реализации RAID 6, написание своей реализации вычислений и сопоставление с результатами коллег для выявления наилучшего алгоритма.

1 ОБЩИЕ ПОЛОЖЕНИЯ

Диски являются самой медленной частью системы. Объединив несколько дисков в один массив, мы можем ускорить работу системы за счет распараллеливания операций обмена данными. При этом с увеличением количества дисков вероятность безотказной работы массива падает, так как она есть произведение вероятностей безотказной работы дисков.

1.1 Определение страйпа и его структура

Основной единицей обработки данных в системе хранения является страйп (Stripe). Страйп разбивается на диски одинакового размера, обозначаемые D0, D1, … , DN-1.

N - количество дисков данных в массиве. Для обеспечения отказоустойчивости в страйп вводятся дополнительные диски, предназначенные для хранения синдромов. Размер этих дополнительных дисков равен размеру дисков данных. Значения синдромов используются в случае потери данных для их восстановления

1.2 Представление дисков

Каждый из дисков данных будем считать элементов поля Галуа $GF(2^n)$, с операцией умножения, заданной по двойному модулю 2 и полиному f(x) являющемуся неприводимым.

Считаем, что количество дисков не превосходит $2^n - 1$ и каждый диск представляет собой n-битный блок (или в альтернативном представлении

полином). Вычисления осуществляем по правилам: суммирование понимается поразрядным, а умножение - по двойному модулю 2, f(x).

Также нам потребуется произвольный примитивный элемент поля. Он используется для вычисления кода Рида-Соломона, о котором будет сказано дальше. Требование примитивности существенно, потому что нам нужно чтобы его степени все были различными и, следовательно, порождали всё наше поле GF(2^n), за исключением нулевого элемента.

- **Теорема 1.** В поле Галуа GF(p^m) существуют первообразные корни степени $p^m - 1$ из единицы.
  В поле GF(p^m) первообразный корень степени $p^m - 1$ из единицы называется примитивным элементом поля.

- **Теорема 2.** Если каноническое разложение числа $p^m - 1$ имеет вид:
  $$p^m - 1 = p_1^{m_1} p_2^{m_2} \times \cdots \times p_r^{m_r} ,$$
  то для того чтобы элемент $\mathfrak{A} \in \mathbf{GF}(p^m)$, был примитивным элементом этого поля, необходимо и достаточно, чтобы $\mathfrak{A}^{(p^m-1)/p_j} \neq \mathfrak{e} \quad npu \quad \forall j \in \{1, \ldots, \mathfrak{r}\}$.

- **Теорема 3.** Если $\mathfrak{A} \in \mathbf{GF}(p^m)$ — примитивный элемент поля, то $\mathfrak{A}^k$ будет примитивным элементом поля тогда и только тогда, когда показатель k взаимно прост с $p^m - 1$: $\mathrm{HOD}(k, p^m - 1) = 1$.

(Доказательства теорем расположены по ссылке №4 из списка вспомогательной литературы)

1.3 Коды Рида-Соломона

Код Рида — Соломона был изобретён в 1960 году сотрудниками лаборатории Линкольна Массачусетского технологического института Ирвином Ридом и Густавом Соломоном. Идея использования этого кода была представлена в статье «Polynomial Codes over Certain Finite Fields». Первое применение код Рида — Соломона получил в 1982 году в серийном выпуске компакт-дисков. Эффективные алгоритмы декодирования были предложены в 1969 году Элвином Берлекэмпом и Джэймсом Месси (алгоритм Берлекэмпа — Мэсси) и в 1977 году Давидом Мандельбаумом (метод, использующий Алгоритм Евклида).

2 СХД УРОВНЯ RAID 6

Существуют различные уровни RAID. Преимуществами RAID 6 по сравнению с RAID 1,2,3,4,5 являются экономичность и одна из самых высоких степеней надежности. Вместе с тем, например, его производительность ниже RAID 5, но RAID 5 не работает так как нужно с информацией, объем которой превышает 5 Tb.

В RAID 6 под контрольные суммы выделяется емкость 2-х дисков, и рассчитываются 2 контрольные суммы по разным алгоритмам. Для организации массива требуется как минимум 4 диска.

2.1 Вычисление синдромов

Контрольные суммы называются синдромами и вычисляются по следующим формулам:

$$P = D_{n-1} + D_{n-2} + \ldots + D_0 = \sum_{i=0}^{n-1} D_i$$

$$Q = x^1 D_{n-1} + x^2 D_{n-2} + x^n D_0 = \sum_{i=0}^{n-1} x^{n-i} D_i$$

Если отказал диск P или Q, то восстанавливать данные не нужно.
В случае утраты диска с номером α, вычислим синдром с пропуском утраченного диска и вычтем результат из полного синдрома P и найдем утраченный диск.
Сделать это мы можем и при помощи синдрома Q.

Для этого храним таблицу предвычисленных значений

- x^{a-n} при $1 \le n - \alpha \le N$. Размерность массива N+1.

1	2	...	n - **α**	...	N - 1	N
x^{-1}	x^{-2}		x^{a-n}	...	$x^{-(N-1)}$	x^{-N}

Для восстановления двух дисков используются оба синдрома P и Q.

Решается линейная система из 2-х уравнений.

Для этого предварительно вычисляются 2 таблицы.

- x^{a-n} при $1 \le n - \alpha \le N$ (таблица предвычисленных значений для восстановления одного диска)
- $(1 - x^{a-B})^{-1}$ при $0 \le \alpha - \beta + N - 1 \le 2N - 2$

0	1	...	$\alpha - \beta + N - 1$	...	N - 2	N - 1	N	...	2N - 2
$1/(1 - x^{1-N})$	$1/(1 - x^{2-N})$	...	$1/(1 - x^{a-B})$	...	$1/(1 - x^{-1})$	$\alpha = \beta$	$1/(1 - x^{1})$	...	$1/(1 - x^{N-1})$

3 АЛГОРИТМИЧЕСКИЕ ОПТИМИЗАЦИИ

На этапе проектирования алгоритма очень важным является создание максимально линейного кода программы, не содержащего каких-либо ветвлений. Это даёт возможность оптимально использовать конвейер команд процессора.

При вычислении значений блоков потерянных данных используется довольно медленная операция деления многочленов. Так как при инициализации системы мы уже знаем максимальное количество дисков в ней, то будет разумно предподсчитать все возможные знаменатели этой формулы. Поиск обратного многочлена рационально проводить алгоритмом Евклида.

3.1 Команды intrinsics и работа с кэшем

Современные процессоры поддерживают массу расширений набора команд, в частности, наборы SSE, SSE2, SSE4. Они очень удобны в использовании и при написании кода на языке C вызываются intrinsics командами, которые позволяют производить дополнительные оптимизации на этапе компиляции кода программы.
Данные расширения наборов команд разработаны для поддержки вычислений в приложениях multimedia, графики и коммуникаций.
Для этих приложений свойственна необходимость производить одинаковые несвязанные вычисления для большого набора данных.
Все команды SSE, кроме специально оговоренных, при работе с памятью требуют выравнивания операнда на границу 16 байт.
В противном случае возникает ошибка переполнения.

В нашей реализации мы храним коэффициенты многочленов в переменных типа _m128i. И оперируем ими, используя команды intrinsics.

3.2 Умножение многочленов

Очень важным оказывается использование команды PCLMULQDQ из набора CLMUL, которая позволяет перемножать 64-битные половины 128-битных переменных типа _m128i (именно поэтому удобно выбирать соответствующие многочлены факторизации и размеры дисковых блоков). При этом умножение в процессоре проходит с выключенными цепями переноса, т.е. при переполнении разряда он сбрасывается на ноль без увеличения ближайшего старшего разряда. Эта команда полностью соответствует описанному в нашем поле умножению и тем самым позволяет гораздо быстрее его выполнять.

4 Замеры производительности

Элементарной единицей замера производительности является тест скорости, запускающий по очереди каждую из пяти операций и измеряющий количество тиков, которое прошло за время её выполнения, т.е. разность между значениями в процессорном счётчике тиков непосредственно перед запуском операции и после её завершения. Текущее состояние счётчика процессорных тиков получалось командой RDTSC().

Описание структуры теста скорости:

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. Случайным образом выбираются два различных номера дисков, для восстановления и перезаписи N1 и N2.

3. Массив A, используемый в качестве новых данных, для пересчёта синдромов P и Q в СХД, размера S заполняется случайными данными.

4. Производятся замеры:

    a. Замеряется количество тиков, требуемое для выполнения функции инициализации RAID (функция 1 из параграфа 2) с параметрами S и D, и сохраняется в K-ой ячейке массива времени исполнения первой операции.

b. Замеряется количество тиков, требуемое для выполнения функции пересчёта синдромов при замене данных в одном из дисков с параметрами S, D, N1 и A, и сохраняется в K-ой ячейке массива времени исполнения второй операции.

c. Замеряется количество тиков, требуемое для выполнения функции восстановления одного диска по синдрому чётности P с параметрами S, D и N1, и сохраняется в K-ой ячейке массива времени исполнения третьей операции.
d. Замеряется количество тиков, требуемое для выполнения функции восстановления одного диска по синдрому поля Галуа Q с параметрами S, D и N1, и сохраняется в K-ой ячейке массива времени исполнения четвёртой операции.

e. Замеряется количество тиков, требуемое для выполнения функции восстановления двух дисков по синдромам P и Q с параметрами S, D, N1, N2, и сохраняется в K-ой ячейке массива времени исполнения пятой операции.

Процедуре замера скорости при запуске передаются три параметра:

- количество жестких дисков в эмулируемой СХД (включая служебные диски, предназначенные для хранения синдромов)
- размер одного диска эмулируемой СХД («ширина» страйпа)
- число элементарных тестов

Элементарные тесты выполняются заданное число раз и для полученных массивов времён исполнения операций считаются значения среднего времени исполнения E для каждой операции и среднеквадратичного отклонения времени исполнения операции от посчитанного среднего

времени исполнения D. Среднее время считается следующим образом: все результаты для одной операции сортируются по возрастанию, далее от полученного отсортированного массива «отщепляется» какая-то часть самых больших и самых маленьких по величине результатов (в проведенных тестах использовалось значение 10, то есть 1/10 верхних и нижних значений откидывались). Проводилось это для того, чтобы предотвратить влияние разнообразных скачков показаний, которые могут быть следствием малопредсказуемых вещей вроде запуска какой-нибудь системной службы или падения напряжения.

5 Результат курсовой работы

В рамках курсовой работы

- **Были изучены алгоритмы расчета RAID 6.**

- **Были изучены наборы процессорных расширений SSE, SSE2, SSE 4.2 и их intrinsics аналоги в языке C, а также способы реализации алгоритмов полиномиального кодирования.**

- **На языке C была написана библиотека подпрограмм для вычислений в поле многочленов с фактор-многочленом $x^{128} + x^7 + x^2 + x + 1$ с использованием intrinsics команд.**

- **Реализован алгоритм RAID 6 с использованием указанного полинома 128-й степени на языке C и Assembler**

Наборы тестовых параметров системы включали в себя количество физических дисков в системе от 4 до 64 с шагом в 4 диска и размеры дисковых страйпов 4 Кбайт и 8 Кбайт. График результатов измерений данной реализации представлен на рис. № 1 Приложения.

Если посмотреть на табл. №1 Приложения, при выполнении всех операций лучший результат показал алгоритм EngineM8R, представляющий собой реализацию матричного алгоритма. Он реализовывался другой группой параллельно. Таким образом, было получено, что матричный алгоритм работает в среднем в 2-3 раза быстрее на большом количестве дисков, чем алгоритм, основанный на полиноме 128 степени.

Стоить отметить, что EngineM8R по свои показателям превосходит также текущую реализацию RAID 6 в СХД AVRORAID - реализацию алгоритма Майка.
Отсюда следует, что наиболее предпочтительным оказался матричный алгоритм, и его следует использовать в реализации СХД уровня RAID 6.

6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Фёдоров А.Р. "Операции с многочленами с двоичными коэффициентами"**
2. **Anvin H.P. "The mathematics of RAID-6"**
3. **Intel® 64 and IA-32 Architectures Software Developer Manuals**
   http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html
4. http://pmpu.ru/vf4/gruppe/galois/vspom2
5. http://pmpu.ru/vf4/codes/raid
6. http://wasm.ru/publist.php?list=11
7. http://www.insidepro.com/kk/141/141r.shtml

ПРИЛОЖЕНИЕ

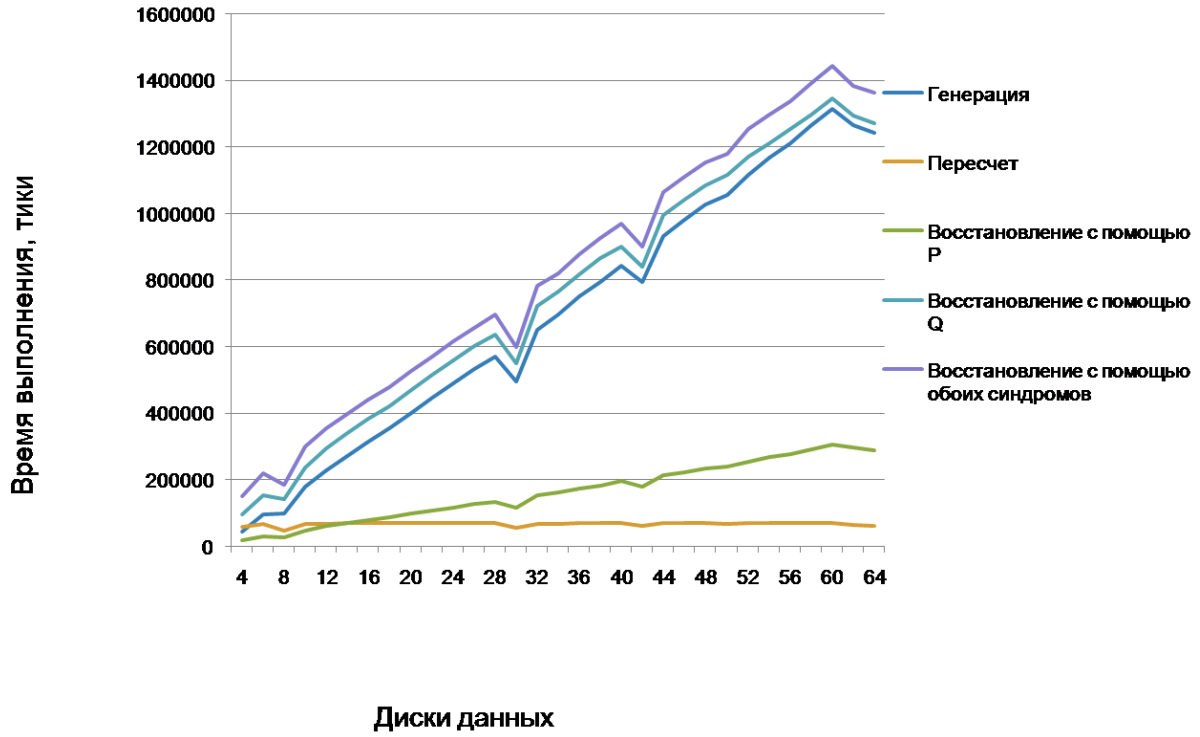

рис.№1

Генерация синдромов			
Диски	Алгоритм Майка	EngineM8R	Engine128asm
8	17151	18413	14099
10	20755	23696	29228
12	25097	29942	46222
14	31420	35401	55263
16	39908	40205	66375
18	50545	46717	88837
20	63778	54930	104653
22	69806	67195	125977

24	77523	75622	144465
Пересчет синдромов			
Диски	Алгоритм Майка	EngineM8R	Engine128asm
8	26646	8148	27663
10	29096	7626	28297
12	29493	7853	42496
14	30270	7982	33008
16	31042	8059	53719
18	30684	8148	56379
20	31159	8329	51515
22	31209	8279	58071
24	30345	8327	73936
Восстановление 2х дисков			
Диски	Алгоритм Майка	EngineM8R	Engine128asm
8	26625	30862	80099
10	29654	37114	96794
12	34313	44059	155901
14	40071	49211	168025
16	47777	54807	186949
18	63036	63350	221566
20	74996	70666	235090
22	80891	85838	283259
24	87917	92246	327877

табл. №1