СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последнее десятилетие центральные процессорные устройства (ЦПУ) достигли своей пиковой тактовой частоты – около 5 Ггц, и этот предел будет преодолён ещё не скоро [1]. Из-за того что ЦПУ стали настолько быстрыми, становится нередка ситуация, когда процессор не выполняет инструкции, а ждёт, пока данные переместятся с диска в оперативное запоминающее устройство (или наоборот) [2]. Так, например, скорость работы системы с мощным ЦПУ, но с маленьким количеством ОЗУ может быть маленькой – несмотря на быстроту, ЦПУ чисто ожидает подсистему ввода/вывода [2]. Существует несколько способов увеличения количества оперативной памяти. Один из способов заключается в физическом увеличении количества планок ОЗУ в системе. Данный способ подразумевает покупку и установку планок ОЗУ, что требует денежных затрат. Кроме физического способа увеличения количества памяти, существуют программные способы увеличения количества ОЗУ, например, сжатие данных. Данный способ требует только вычислительные мощности ЦПУ [3], а как было указано ранее, ЦПУ часто простаивает в ожидании операций ввода/вывода. Данный факт позволяет направить простаивающие вычислительные мощности ЦПУ на обработку операций сжатия оперативной памяти.

Цель работы – разработка оптимизации метода сжатия страниц оперативной памяти в ядре Linux.

Для достижения поставленной цели необходимо решить следующие задачи:

- спроектировать структуру программного обеспечения, реализующего оптимизацию модуля сжатия оперативной памяти;
- разработать программное обеспечение для данной оптимизации;
- исследовать разработанное ПО: сравнить метод сжатия страниц оперативной памяти с оптимизацией и без.

1 Программная реализация оптимизации метода сжатия страниц оперативной памяти в ядре Linux

В данном разделе описываются средства разработки программного обеспечения и требования к нему. Приводится структура разработанного ПО.

1.1 Выбор средств разработки

1.1.1 Выбор языка программирования

Ядро ОС Linux написано на языке программирования C с использованием стандарта C89. Встроенный драйвер ядра Linux, zram, так же написан на языке программирования C. Программное обеспечение представляет из себя модификацию модуля ядра zram, в связи с чем для реализации программного обеспечения был выбран язык программирования C версии стандарта C89. В модуле ядра zram, как и в ядре Linux, используется процедурный подход к программированию. Язык C поддерживает процедурную парадигму программирования.

1.1.2 Версия ядра Linux

В качестве версии ядра Linux была выбрана версия 5.17.5. На момент написания дипломной работы, эта версия является самой актуальной версией ядра Linux. Данный факт позволяет использовать все возможности ядра при разработке программного обеспечения.

1.2 Сборка программного обеспечения

Разработанное программное обеспечение является часть ядра Linux. Для сборки проекта используется специальная утилита make, позволяющая автоматизировать сборку ядра. make является кроссплатформенной системой автоматизации сборки программного обеспечения из исходного кода. make позволяет существенно ускорить процесс сборки проекта. Так, например, при изменении одного исходного файла проекта, заново будет собран в объектный файл лишь этот исходный файл, а не все файлы проекта.

Для сборки и включения модуля zram в итоговый образ ядра Linux, в конфигурационном файле ядра необходимо включить опции, указанные в листинге

1. Ниже приведено детальное описание включаемых опций:

- `CONFIG_ZRAM` – опция, включающая поддержку модуля zram в итоговый образ ядра Linux. Без этой опции, использовать zram будет невозможно;
- `CONFIG_ZRAM_DEF_COMP_ZSTD` и `CONFIG_ZRAM_DEF_COMP="zstd"` – выбрать алгоритм сжатия, который будет использоваться при сжатии страниц оперативной памяти. В данном случае, выбран алгоритм сжатия zstd;
- `CONFIG_ZRAM_ENTROPY` – эта опция включает поддержку энтропийной оптимизации модуля zram.
- `CONFIG_ZRAM_ENTROPY_THRESHOLD=100000` – данный параметр задает границу для отсекаемых страниц. Страницы, энтропия которых выше данного значения, будут храниться не в сжатом виде. Данное значение для каждого алгоритма сжатия выбирается оптимальным образом автоматически, но, при этом, имеется возможность установить его в ручную.

Разработанное программное обеспечение представлено в виде опции. Эту опцию можно выключать и отключать на стадии сборки ядра.

Листинг 1: Опции, которые необходимо добавить в конфигурационный файл ядра

1.3 Требования к вычислительной системе

Разработанное программное обеспечение представляет из себя модификацию ядра Linux. Для сборки и установки ядра в систему требуются следующие библиотеки и утилиты, представленные в таблице 1.

1.4 Структура программного обеспечения

Разработанное ПО представляет из себя функцию вычисления информационной энтропии, её вызов перед попыткой сжатия страницы и сравнение с пороговым значением. Ниже описывается эта функция и модификация функции, в которой происходит её вызов и принятие решение о дальнейшем хранении страницы памяти.

Таблица 1 – Таблица ПО, необходимого для сборки и установки ядра

ПО	**Минимальная версия**
gcc	3.2
GNU make	3.80
binutils	2.12
util-linux	2.10o
module-init-tools	0.9.10
e2fsprogs	1.41.4
jfsutils	1.1.3
reiserfsprogs	3.6.3
xfsprogs	2.6.0
squashfs-tools	4.0
btrfs-progs	0.18
pcmciautils	004
quota-tools	3.09
PPP	2.4.0
isdn4k-utils	3.1pre1
nfs-utils	1.0.5
procps	3.2.0
oprofile	0.9
udev	081
grub	0.93
mcelog	0.6
iptables	1.4.2
openssl, libcrypto	1.0.0
bc	1.2

1.4.1 Функция вычисления информационной энтропии

Функция shannon_entropy в качестве единственного входного параметра получает указатель на массив байт размером PAGE_SIZE. Возвращает информационную энтропию, подсчитанную по формуле 1, для данного набора байт.

$$H(x) = -K \sum_{i=1}^{n} p(x_i) log_2 p(x_i), \tag{1}$$

где $p(x_i)$ – вероятность i-го состояния системы (значения принимаемого переменной), n – число состояний системы (значений, принимаемых переменной), K – положительная константа.

Для вычисления логарифма используется встроенная функция ядра ilog2. В ядре Linux нет поддержки чисел с плавающей запятой, поэтому функция ilog2 работает только с целыми числами. Передаваемый параметр в функцию ilog2 (p_i) возводится в 4 степень для увлечения точности вычислений.

Пример реализации функции shannon_entropy() для модуля zram представлен в листинге 2.

Листинг 2: Функция shannon_entropy()

```
static inline u32 ilog2_w(u64 n)
{
    return ilog2(n * n * n * n);
}

static inline s32 shannon_entropy(const u8 *src)
{
    s32 entropy_sum = 0;
    u32 sz_base, i;
    u16 entropy_count[256] = { 0 };

    for (i = 0; i < PAGE_SIZE; ++i)
        entropy_count[src[i]]++;

    sz_base = ilog2_w(PAGE_SIZE);
    for (i = 0; i < ARRAY_SIZE(entropy_count); ++i) {
        if (entropy_count[i] > 0) {
            s32 p = entropy_count[i];

            entropy_sum += p * (sz_base - ilog2_w((u64)p));
        }
    }

    return entropy_sum;
}
```

1.4.2 Принятие решение о дальнейшем хранении страницы

Сжатие страницы происходит в функции zcomp_compress(), которая, в свою очередь вызывается в функции __zram_bvec_write(). Вызов функции

shannon_entropy() необходимо произвести до сжатия данных, хранящихся на странице памяти, поэтому функция __zram_bvec_write() была модифицирована.

Перед вызовом zcomp_compress() для каждой попавшей в эту функцию страницы считается информационная энтропия. Если вычисленное значение выше порогового значения CONFIG_ZRAM_ENTROPY_THRESHOLD, объявленного с помощью директивы #define, то эта страница помечается как несжимаемая в памяти. В обратном случае, происходит вызов zcomp_compress() и данные, находящиеся на этой странице памяти, сжимаются.

Пример реализации модификации функции __zram_bvec_write() модуля zram представлен в листинге 3.

Листинг 3: Функция __zram_bvec_write()

```
static int __zram_bvec_write(struct zram *zram,
    struct bio_vec *bvec,
    u32 index,
    struct bio *bio) {
...
    zstrm = zcomp_stream_get(zram->comp);
    src = kmap_atomic(page);

#ifdef CONFIG_ZRAM_ENTROPY
    entropy = shannon_entropy((const u8 *)src);
    if (entropy > CONFIG_ZRAM_ENTROPY_THRESHOLD)
        comp_len = PAGE_SIZE;
    else
        ret = zcomp_compress(zstrm, src, &comp_len);
#else
    ret = zcomp_compress(zstrm, src, &comp_len);
#endif

    kunmap_atomic(src);
...
}
```

1.5 Руководство пользователя

Для запуска разработанного ПО необходимо запущенное ядро Linux, собранное с опциями указанными в главе 3.3. Список включенных опций на уже запущенном ядре Linux можно узнать с помощью команды zcat /proc/config.gz/.

Собрать ядро Linux из исходных файлов можно с помощью команды make -j$(nproc). До сборки необходимо убедиться, что в конфигурационном файле,

имеющем имя .config и находящемся в корне проекта, включены опции указанные в главе 3.2. Для того чтобы загрузчик мог использовать собранный образ ядра, необходимо установить его с помощью команд make modules_install и make install.

Для того чтобы инициализировать и задать размер блочного устройства zram необходимо использовать команду: echo 512M > /sys/block/zram0/disksize. В данном примере размер устройства равен 512 мегабайтам. Задавать размер, превышающий размер ОЗУ более чем в 2 раза не имеет смысла: в среднем коэффициент сжатия равен двум [4].

Для того чтобы установить блочное устройство zram в качестве устройства, которое будет использовать в подкачке страниц, необходимо использовать следующие команды: mkswap /dev/zram0 и swapon /dev/zram0.

1.6 Вывод

В данном разделе были описаны средства разработки программного обеспечения и требования к ПО. Была приведена структура разработанного ПО.

2 Исследование разработанной оптимизации метода сжатия страниц оперативной памяти в ядре Linux

В рамках дипломной работы было проведено исследование сравнения времени обработки и коэффициент сжатия файлов в блочном устройстве zram с оптимизацией и без. Результаты исследования представлены в данном разделе.

2.1 Описание используемых данных

Для исследования работоспособности разработанного программного обеспечения и оценки времени обработки данных и коэффициента их сжатия были выбраны бинарные файлы различного типа и разного размера. В исследовании использовались файлы типа pdf (portable document format [5]), apk (android package [6]) и случайные файлы из домашней директории, сохраненные в единый файл с помощью утилиты tar [7]. Размер файлов составляет 400 мегабайт, 430 мегабайт и 5 гигабайт соответственно.

2.2 Методика проведения исследования

Для исследования необходимо произвести замеры количества машинных инструкций, времени сжатия и коэффициента сжатия данных с использованием энтропийной оптимизации и без. Количество машинных инструкций и время выполнения подсчитывается с помощью утилиты perf [8], а коэффициент сжатия данных с использованием встроенной в модуль zram статистики.

В таблице 2 представлены результаты сравнения коэффициентов сжатия с включенной разработанной оптимизацией и без. В первом столбце указано, включена энтропийная оптимизация или нет. В ячейках таблицы указан исходный размер данных, сжатый и коэффициент сжатия.

В таблице 3 представлено сравнение времени выполнения и количества машинных инструкций с включенной оптимизацией и без.

2.3 Вывод

В результате исследования было установлено что коэффициент сжатия сильно зависит от входных данных. Так, например, файл формата pdf практи-

Таблица 2 – Таблица сравнения коэффициентов сжатия с оптимизаций и без

Патч	**Файл**	**Размер на входе, кб**	**Размер на выходе, кб**	**Коэф. сжатия**
Да	pdf	397.464	396.535	1.002
Нет	pdf	397.464	395.870	1.004
Да	apk	421.340	327.992	1.284
Нет	apk	421.340	282.331	1.492
Да	tar	5.153.692	4.131.741	1.247
Нет	tar	5.153.692	4.117.655	1.251

Таблица 3 – Таблица сравнения количества машинных инструкций с оптимизаций и без

Патч	**Файл**	**Размер на входе, кб**	**Время сжатия, с**	**Кол-во инструкций**
Да	pdf	397.464	0.572	4.641.658.347
Нет	pdf	397.464	2.098	17.187.405.320
Да	apk	421.340	1.632	13.055.231.312
Нет	apk	421.340	3.196	24.345.720.313
Да	tar	5.153.692	7.066	35.039.839.769
Нет	tar	5.153.692	11.955	76.763.123.464

чески не сжался как с включенной энтропийной оптимизацией, так и без.

Оптимизация метода сжатия ускоряет процесс преобразования данных в среднем от 2 до 4 раз. Чем меньше процент сжатия исходных данных без включенной оптимизации, тем больше ускоряется процесс сжатия с включенной оптимизацией. Файл формата pdf потерял маленькую долю сжатия (коэффициент сжатия стал 1.002, вместо 1.004), но при этом, процесс преобразования данных ускорился в 4 раза.

Разносортные данные упакованные в единый файл сжимаются в среднем

на 25%. Сжатие с энтропийной оптимизацией происходит в 1.9 раза быстрее, чем без. При этом, потеря в сжатии составляет менее процента.

Подводя итог, можно сделать вывод, что разработанная оптимизация программного обеспечения ускоряет процесс сжатия в несколько раз (2-4 раза), при этом потеря в сжатии не крайне малы: от 1% до 14%, в зависимости от входных данных.

Полученные результаты, вместе с модификацией модуля zram, были отправлены в качестве RFC (англ. request for comments [9]) письма мейнтейнерам модуля ядра zram [10].

ЗАКЛЮЧЕНИЕ

Были описаны средства разработки программного обеспечения и требования к нему и приведена структура разработанного ПО.

Было проведено исследование сравнения времени обработки и коэффициент сжатия файлов в блочном устройстве zram с оптимизацией и без.

Таким образом, цель работы – разработать оптимизацию метода сжатия страниц оперативной памяти, была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Why Aren’t CPUs Getting Faster? - Apple Gazette [Электронный ресурс]. – Режим доступа: https://applegazette.com/mac/why-arent-cpus-getting-faster/, свободный – (10.11.2021)

2. In-kernel memory compression [Электронный ресурс]. – Режим доступа: https://lwn.net/Articles/545244/, свободный – (10.11.2021)

3. Data compression | computing - Encyclopedia Britannica [Электронный ресурс]. – Режим доступа: https://www.britannica.com/technology/data-compression, свободный – (24.03.2021)

4. zram: Compressed RAM-based block devices - The Linux Kernel Documentation [Электронный ресурс]. – Режим доступа: https://www.kernel.org/doc/html/latest/admin-guide/blockdev/zram.html, свободный – (10.11.2021)

5. What is a PDF? Portable Document Format | Adobe Acrobat [Электронный ресурс]. – Режим доступа: https://www.adobe.com/acrobat/about-adobe-pdf.html, свободный – (27.04.2022)

6. What is APK file (Android Package Kit file format)? - TechTarget [Электронный ресурс]. – Режим доступа: https://www.techtarget.com/whatis/definition/APK-file-Android-Package-Kit-file-format, свободный – (27.04.2022)

7. tar(1) - Linux manual page - man7.org [Электронный ресурс]. – Режим доступа: https://man7.org/linux/man-pages/man1/tar.1.html, свободный – (27.04.2022)

8. Perf Wiki | Linux Kernel [Электронный ресурс]. – Режим доступа: https://perf.wiki.kernel.org/index.php/Main_Page, свободный – (27.04.2022)

9. PatchTipsAndTricks - Linux Kernel Newbies [Электронный ресурс]. –

Режим доступа: https://kernelnewbies.org/PatchTipsAndTricks, свободный – (27.04.2022)

10. [RFC,v1] zram: experimental patch with entropy calculation | Patchwork [Электронный ресурс]. – Режим доступа: https://patchwork.kernel.org/project/linux-block/patch/20220520171309.26768-1-avromanov@sberdevices.ru/, свободный – (27.04.2022)