Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.01 Программная инженерия

Дисциплина «Операционные системы»

Отчет

По лабораторной работе №1

Вариант - stress-ng

Выполнил:

Кадилов М. В.

Преподаватель:

Зубахин Д. С.

Санкт-Петербург, 2025 г.

Оглавление

Задание

Лабораторная работа №1

Задание Часть 1. Запуск программ

Необходимо реализовать собственную оболочку командной строки - shell. Выбор ОС для реализации производится на усмотрение студента. Shell должен предоставлять пользователю возможность запускать программы на компьютере с переданными аргументами командной строки и после завершения программы показывать реальное время ее работы (подсчитать самостоятельно как «время завершения» – «время запуска»).

Задание Часть 2. Мониторинг и профилирование

Разработать комплекс программ-нагрузчиков по варианту, заданному преподавателем. Каждый нагрузчик должен, как минимум, принимать параметр, который определяет количество повторений для алгоритма, указанного в задании. Программы должны нагружать вычислительную систему, дисковую подсистему или обе подсистемы сразу. Необходимо скомпилировать их без опций оптимизации компилятора.

Перед запуском нагрузчика, попробуйте оценить время работы вашей программы или ее результаты (если по варианту вам досталось измерение чего либо). Постарайтесь обосновать свои предположения. Предположение можно сделать, основываясь на свой опыт, знания ОС и характеристики используемого аппаратного обеспечения.

1. Запустите программу-нагрузчик и зафиксируйте метрики ее работы с помощью инструментов для профилирования. Сравните полученные результаты с ожидаемыми. Постарайтесь найти объяснение наблюдаемому.
2. Определите количество нагрузчиков, которое эффективно нагружает все ядра процессора на вашей системе. Как распределяются времена USER%, SYS%, WAIT%, а также реальное время выполнения нагрузчика, какое количество переключений контекста (вынужденных и невынужденных) происходит при этом?
3. Увеличьте количество нагрузчиков вдвое, втрое, вчетверо. Как изменились времена, указанные на предыдущем шаге? Как ведет себя ваша система?
4. Объедините программы-нагрузчики в одну, реализованную при помощи потоков выполнения, чтобы один нагрузчик эффективно нагружал все ядра вашей системы. Как изменились времена для того же объема вычислений? Запустите одну, две, три таких программы.
5. Добавьте опции агрессивной оптимизации для компилятора. Как изменились времена? На сколько сократилось реальное время исполнения программы нагрузчика?

Ограничения

Программа (комплекс программ) должна быть реализован на языке C, C++. Дочерние процессы должны быть созданы через заданные системные вызовы выбранной операционной системы, с обеспечением корректного запуска и и завершения процессов. Запрещено использовать высокоуровневые абстракции над системными вызовами. Необходимо использовать, в случае Unix, процедуры libc.

Требования к отчету и защите

- Отчет должен содержать титульный лист с указанием номера и названия ЛР, вашего ФИО, ФИО преподавателя практики, номера вашей группы, варианта ЛР.
- Отчет должен содержать текст задания в соответствии с вариантом.
- Отчет должен содержать листинг исходного кода всех программ, написанных в рамках данной ЛР.
- Отчет должен содержать предположения о свойствах программ-нагрузчиков
- Отчет должен содержать результаты измерений и метрик программ-нагрузчиков, полученных инструментами мониторинга. Должно быть описано, какие утилиты запускались, с какими параметрами и выводом.
- Отчет должен содержать сравнительный анализ ожидаемых и фактических значений.
- Отчет должен содержать вывод.
- Студент должен быть готов продемонстрировать работоспособность Shell и предоставить исходный код.
- Студент должен быть готов воспроизвести ход работы в рамках части 2 и продемонстрировать схожие результаты работы программ-нагрузчиков.

Темы для подготовки к защите лабораторной работы:

1. Структура процесса и потоков;
2. Системные утилиты сбора статистики ядра;
3. Основы ввода-вывода (блочный и последовательный ввод-вывод);
4. Файловая система procfs;
5. Использование утилиты strace, ltrace, bpftrace;
6. (*) Профилирование и построение flamegraph'а и stap;

Вариант

Кадилов Михаил Владимирович stress-ng (--icache 15 --far-branch-flush --timeout 2h, --flushcache 15 --far-branch-flush --timeout 2h, --cache 10 --cache-enable-all --timeout 2h,--branch 10 --far-branch-flush --timeout 2h)

Часть 1

Реализация:

```
#include <bits/stdc++.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/resource.h>
#include <signal.h>
#include <time.h>

using namespace std;

// чистим зомби-процессы
static void sigchld_handler(int) {
    while (waitpid(-1, nullptr, WNOHANG) > 0) {}
}

vector<string> split_tokens(const string &line) {
    vector<string> toks;
    istringstream iss(line);
    string t;
    while (iss >> t) toks.push_back(t);
    return toks;
}

int main() {
    // ставим обработчик SIGCHLD, чтобы не плодить зомби
    struct sigaction sa{};
    sa.sa_handler = sigchld_handler;
```

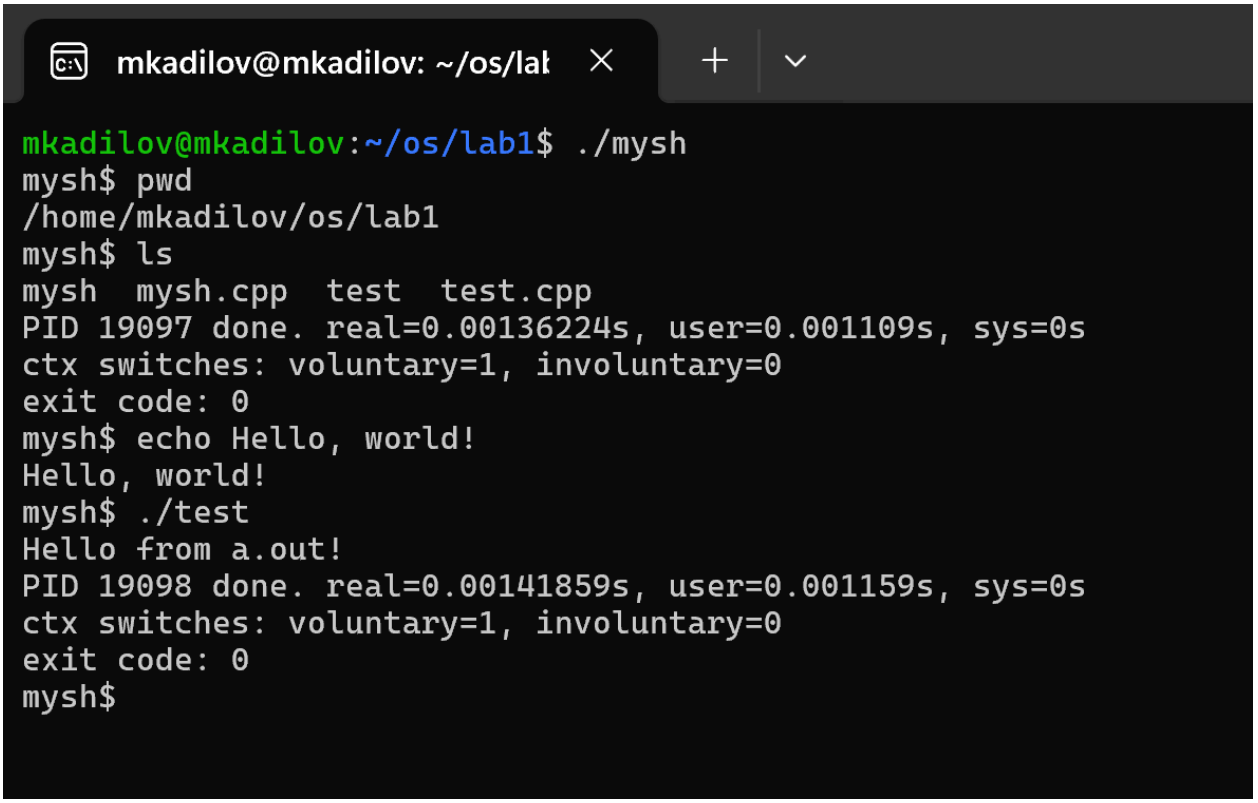
```
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = SA_RESTART | SA_NOCLDSTOP;
    sigaction(SIGCHLD, &sa, nullptr);


    string line;
    while (true) {
        cout << "mysh$ " << flush;
        if (!getline(cin, line)) break;
        if (line.empty()) continue;


        auto toks = split_tokens(line);
        if (toks.empty()) continue;


        // встроенные команды
        if (toks[0] == "exit") {
            cout << "Bye!\n";
            break;
        }


        if (toks[0] == "pwd") {
            char cwd[1024];
            if (getcwd(cwd, sizeof(cwd)) != nullptr)
                cout << cwd << "\n";
            else
                perror("pwd");
            continue;
        }


        if (toks[0] == "cd") {
```

```
            const char* dir = toks.size() > 1 ? toks[1].c_str() :
getenv("HOME");
            if (chdir(dir) != 0) perror("cd");
            continue;
        }

        if (toks[0] == "echo") {
            for (size_t i = 1; i < toks.size(); ++i)
                cout << toks[i] << (i + 1 < toks.size() ? " " : "");
            cout << "\n";
            continue;
        }

        bool background = false;
        if (!toks.empty() && toks.back() == "&") {
            background = true;
            toks.pop_back();
            if (toks.empty()) continue;
        }

        vector<char*> argv;
        for (auto &s : toks) argv.push_back(const_cast<char*>(s.c_str()));
        argv.push_back(nullptr);

        struct timespec tstart{}, tend{};
        clock_gettime(CLOCK_MONOTONIC, &tstart);

        pid_t pid = fork();
        if (pid < 0) {
            perror("fork");
```

```
            continue;
        }

        if (pid == 0) {
            execvp(argv[0], argv.data());
            perror("execvp");
            _exit(127);
        } else {
            if (background) {
                cout << "[bg] PID " << pid << "\n";
                continue;
            }

            int status = 0;
            struct rusage usage{};
            wait4(pid, &status, 0, &usage);
            clock_gettime(CLOCK_MONOTONIC, &tend);

            double real = (tend.tv_sec - tstart.tv_sec)
                        + (tend.tv_nsec - tstart.tv_nsec)/1e9;
            double user = usage.ru_utime.tv_sec +
usage.ru_utime.tv_usec/1e6;
            double sys  = usage.ru_stime.tv_sec +
usage.ru_stime.tv_usec/1e6;

            cout << "PID " << pid << " done. real=" << real
                 << "s, user=" << user << "s, sys=" << sys << "s\n";
            cout << "ctx switches: voluntary=" << usage.ru_nvcsw
                 << ", involuntary=" << usage.ru_nivcsw << "\n";

            if (WIFEXITED(status))
```

```
                cout << "exit code: " << WEXITSTATUS(status) << "\n";
            else if (WIFSIGNALED(status))
                cout << "killed by signal: " << WTERMSIG(status) << "\n";
        }
    }

    return 0;
}
```

Результаты запуска реализации shell:



Часть 2

Теоретические предположения перед запуском

Перед запуском программы-нагрузчика предполагалось, что при увеличении количества потоков (параметра `--icache`, `--flushcache`, `--cache`, `--branch`) произойдёт рост нагрузки на процессор, увеличение доли времени в режиме SYS% и количества переключений контекста. Это объясняется тем, что каждый поток создаёт дополнительные системные вызовы для управления синхронизацией, планирования и доступа к ресурсам операционной системы. Также параметры, такие как `--icache`, `--flushcache`, `--cache` и `--branch`, приводят к дополнительным операциям ядра для работы с кэшами и ветвлениями. В результате увеличивается доля времени, которое процессор тратит на выполнение функций ядра, а не пользовательского кода. При этом при полном использовании всех ядер ожидалось, что CPU Load достигнет ~800% (при 8 ядрах).

Тест 1: Instruction Cache (--icache)

1.1. Тест на 10 секунд

Команда для запуска:

/usr/bin/time -v stress-ng --icache 2 --far-branch-flush --timeout 10s

Результаты запуска:

mysh$ /usr/bin/time -v stress-ng --icache 2 --far-branch-flush --timeout 10s

stress-ng: info: [19180] setting to a 10 secs run per stressor

stress-ng: info: [19180] dispatching hogs: 2 icache

stress-ng: info: [19180] skipped: 0

stress-ng: info: [19180] passed: 2: icache (2)

stress-ng: info: [19180] failed: 0

stress-ng: info: [19180] metrics untrustworthy: 0

stress-ng: info: [19180] successful run completed in 10.00 secs

Command being timed: "stress-ng --icache 2 --far-branch-flush --timeout 10s"

User time (seconds): 5.58

System time (seconds): 14.52

Percent of CPU this job got: 198%

Elapsed (wall clock) time (h:mm:ss or m:ss): 0:10.12

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 18396

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 95

Minor (reclaiming a frame) page faults: 3449

Voluntary context switches: 48

Involuntary context switches: 11

Swaps: 0

File system inputs: 10080

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

PID 19179 done. real=10.1366s, user=5.58974s, sys=14.5293s

ctx switches: voluntary=51, involuntary=11

exit code: 0

1.2. Тест на 2 часа

Команда для запуска:

mysh$ /usr/bin/time -v stress-ng --icache 15 --far-branch-flush --timeout 2h

Результаты запуска:

mysh$ /usr/bin/time -v stress-ng --icache 15 --far-branch-flush --timeout 2h

stress-ng: info: [20148] setting to a 2 hours run per stressor

stress-ng: info: [20148] dispatching hogs: 15 icache

stress-ng: info: [20148] skipped: 0

stress-ng: info: [20148] passed: 15: icache (15)

stress-ng: info: [20148] failed: 0

stress-ng: info: [20148] metrics untrustworthy: 0

stress-ng: info: [20148] successful run completed in 2 hours

Command being timed: "stress-ng --icache 15 --far-branch-flush --timeout 2h"

User time (seconds): 16711.44

System time (seconds): 40879.48

Percent of CPU this job got: 799%

Elapsed (wall clock) time (h:mm:ss or m:ss): 2:00:00

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 18480

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 66

Minor (reclaiming a frame) page faults: 4194

Voluntary context switches: 52

Involuntary context switches: 2581132

Swaps: 0

File system inputs: 0

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

PID 20147 done. real=7200.15s, user=16711.5s, sys=40879.5s

ctx switches: voluntary=53, involuntary=2581132

exit code: 0

Сравнительный анализ теста 1: Instruction Cache (--icache)

Сравнение краткосрочного и длительного теста:

Параметр	10 секунд	2 часа	Комментарий
User time	5.58 s	16711.44 s	При увеличении числа стрессоров и времени наблюдается пропорциональный рост времени CPU в user mode
System time	14.52 s	40879.48 s	Системное время растёт быстрее, чем пользовательское, что связано с частыми переключениями и обработкой системных вызовов
% CPU	198%	799%	Рост нагрузки с увеличением числа потоков: на коротком тесте задействовано ~2 ядра, на длинном – все 8 ядер, суммарная загрузка отражает суммарную нагрузку всех CPU-bound потоков.
Elapsed (real)	10.12 s	7200 s	Соответствует заявленному времени выполнения stress-ng
Voluntary context switches	51	53	Почти не изменились, т.к. CPU-bound потоки редко блокируются и не ждут ресурсов.
Involuntary context switches	11	2 581 132	Значительное увеличение, связано с планировщиком ОС при высокой загрузке всех ядер
Page faults	Minor: 3 449, Major: 95	Minor:4 194, Major: 66	Незначительное влияние на производительность

Вывод:

Нагрузчик хорошо масштабируется по количеству стрессоров. Основной рост времени CPU приходится на sys time из-за управления кэшем и системных операций. Реальное время выполнения соответствует ожиданиям. Количество involuntary context switches резко растёт при длительной нагрузке на все ядра.

Тест 2: Flush Cache (`--flushcache`)

2.1. Тест на 10 секунд

Команда для запуска:

mysh$ /usr/bin/time -v stress-ng --flushcache 2 --far-branch-flush --timeout 10s

Результаты запуска:

mysh$ /usr/bin/time -v stress-ng --flushcache 2 --far-branch-flush --timeout 10s

stress-ng: info: [19189] setting to a 10 secs run per stressor

stress-ng: info: [19189] dispatching hogs: 2 flushcache

stress-ng: info: [19189] skipped: 0

stress-ng: info: [19189] passed: 2: flushcache (2)

stress-ng: info: [19189] failed: 0

stress-ng: info: [19189] metrics untrustworthy: 0

stress-ng: info: [19189] successful run completed in 10.01 secs

Command being timed: "stress-ng --flushcache 2 --far-branch-flush --timeout 10s"

User time (seconds): 19.87

System time (seconds): 0.17

Percent of CPU this job got: 197%

Elapsed (wall clock) time (h:mm:ss or m:ss): 0:10.14

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 18324

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 101

Minor (reclaiming a frame) page faults: 9693

Voluntary context switches: 66

Involuntary context switches: 8

Swaps: 0

File system inputs: 10896

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

PID 19188 done. real=10.1529s, user=19.8759s, sys=0.178597s

ctx switches: voluntary=72, involuntary=8

exit code: 0

2.2. Тест на 2 часа

Команда для запуска:

mysh$ /usr/bin/time -v stress-ng --flushcache 15 --far-branch-flush --timeout 2h

Результаты запуска:

mysh$ /usr/bin/time -v stress-ng --flushcache 15 --far-branch-flush --timeout 2h

stress-ng: info: [19514] setting to a 2 hours run per stressor

stress-ng: info: [19514] dispatching hogs: 15 flushcache

stress-ng: info: [19514] skipped: 0

stress-ng: info: [19514] passed: 15: flushcache (15)

stress-ng: info: [19514] failed: 0

stress-ng: info: [19514] metrics untrustworthy: 0

stress-ng: info: [19514] successful run completed in 2 hours

Command being timed: "stress-ng --flushcache 15 --far-branch-flush --timeout 2h"

User time (seconds): 57547.74

System time (seconds): 38.09

Percent of CPU this job got: 799%

Elapsed (wall clock) time (h:mm:ss or m:ss): 2:00:00

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 17788

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 68

Minor (reclaiming a frame) page faults: 51233

Voluntary context switches: 94

Involuntary context switches: 2582849

Swaps: 0

File system inputs: 160

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

PID 19513 done. real=7200.11s, user=57547.7s, sys=38.0987s

ctx switches: voluntary=96, involuntary=2582849

exit code: 0

Сравнительный анализ теста 2: Flush Cache (--flushcache)

Параметр	10 секунд	2 часа	Комментарий
User time	19.87 s	57 547.74 s	User time увеличивается почти линейно с количеством стрессоров
System time	0.17 s	38.09 s	Системное время незначительно по сравнению с user time
% CPU	197%	799%	Нагрузка увеличилась пропорционально числу стрессоров
Elapsed (real)	10.14 s	7200 s	Время выполнения соответствует таймауту
Voluntary context switches	72	96	Лёгкое увеличение, связано с планировщиком
Involuntary context switches	8	2 582 849	Рост при полной загрузке процессора
Page faults	Minor: 9 693, Major: 101	Minor: 51 233, Major: 68	Значительное увеличение minor page faults при длительном тесте

Вывод:

Flush Cache нагрузчик сильно загружает CPU и создаёт большое количество involuntary context switches при длительном тесте, что соответствует ожиданиям для операций с кэшем. Системное время остаётся маленьким, так как операции преимущественно CPU-bound.

Тест 3: Cache (--cache)

3.1. Тест на 10 секунд

Команда для запуска:

mysh$ /usr/bin/time -v stress-ng --cache 2 --cache-enable-all --timeout 10s

Результаты запуска:

mysh$ /usr/bin/time -v stress-ng --cache 2 --cache-enable-all --timeout 10s

stress-ng: info: [19197] setting to a 10 secs run per stressor

stress-ng: info: [19197] dispatching hogs: 2 cache

stress-ng: info: [19198] cache: cache flags used: prefetch flush fence sfence clflushopt clwb prefetchw

stress-ng: info: [19198] cache: unavailable unused cache flags: cldemote

stress-ng: info: [19198] cache: 128 permutations of cache flags being exercised

stress-ng: info: [19197] skipped: 0

stress-ng: info: [19197] passed: 2: cache (2)

stress-ng: info: [19197] failed: 0

stress-ng: info: [19197] metrics untrustworthy: 0

stress-ng: info: [19197] successful run completed in 10.01 secs

Command being timed: "stress-ng --cache 2 --cache-enable-all --timeout 10s"

User time (seconds): 10.44

System time (seconds): 0.03

Percent of CPU this job got: 104%

Elapsed (wall clock) time (h:mm:ss or m:ss): 0:10.05

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 18568

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 3175

Minor (reclaiming a frame) page faults: 6542

Voluntary context switches: 158

Involuntary context switches: 482

Swaps: 0

File system inputs: 11080

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

PID 19196 done. real=10.0693s, user=10.4438s, sys=0.042933s

ctx switches: voluntary=164, involuntary=482

exit code: 0

3.2. Тест на 2 часа

Команда для запуска:

mysh$ /usr/bin/time -v stress-ng --cache 10 --cache-enable-all --timeout 2h

Результаты запуска:

mysh$ /usr/bin/time -v stress-ng --cache 10 --cache-enable-all --timeout 2h

stress-ng: info: [19611] setting to a 2 hours run per stressor

stress-ng: info: [19611] dispatching hogs: 10 cache

stress-ng: info: [19612] cache: cache flags used: prefetch flush fence sfence clflushopt clwb prefetchw

stress-ng: info: [19612] cache: unavailable unused cache flags: cldemote

stress-ng: info: [19612] cache: 128 permutations of cache flags being exercised

stress-ng: info: [19611] skipped: 0

stress-ng: info: [19611] passed: 10: cache (10)

stress-ng: info: [19611] failed: 0

stress-ng: info: [19611] metrics untrustworthy: 0

stress-ng: info: [19611] successful run completed in 2 hours

Command being timed: "stress-ng --cache 10 --cache-enable-all --timeout 2h"

User time (seconds): 28758.79

System time (seconds): 2.03

Percent of CPU this job got: 399%

Elapsed (wall clock) time (h:mm:ss or m:ss): 2:00:00

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 18384

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 3186

Minor (reclaiming a frame) page faults: 31682

Voluntary context switches: 52570

Involuntary context switches: 1334329

Swaps: 0

File system inputs: 12512

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

PID 19610 done. real=7200.15s, user=28758.8s, sys=2.0402s

ctx switches: voluntary=52579, involuntary=1334329

exit code: 0

Сравнительный анализ теста 3: Cache (--cache)

Параметр	10 секунд	2 часа	Комментарий
User time	10.44 s	28 758.79 s	User time увеличивается пропорционально числу стрессоров и времени
System time	0.03 s	2.03 s	Практически незначительно
% CPU	104%	399%	Рост CPU% соответствует увеличению количества стрессоров
Elapsed (real)	10.05 s	7200 s	Реальное время соответствует таймауту
Voluntary context switches	164	52 579	Существенный рост при длительном тесте, система активно планирует потоки
Involuntary context switches	482	1 334 329	Рост из-за полной загрузки CPU и работы с кэшем
Page faults	Minor: 6 542, Major: 3 175	Minor: 31 682, Major: 3 186	Minor page faults растут, Major почти не меняются

Вывод:

Cache стрессор менее системозависим, чем flushcache и icache, большая часть времени уходит на пользовательский CPU, sys time минимален. Количество involuntary context switches увеличивается при длительной нагрузке. Поведение соответствует ожиданиям для кэша с активным включением всех флагов.

Тест 4: Branch Predictor (`--branch`)

4.1. Тест на 10 секунд

Команда для запуска:

mysh$ /usr/bin/time -v stress-ng --branch 2 --far-branch-flush --timeout 10s

Результаты запуска:

mysh$ /usr/bin/time -v stress-ng --branch 2 --far-branch-flush --timeout 10s

stress-ng: info: [19203] setting to a 10 secs run per stressor

stress-ng: info: [19203] dispatching hogs: 2 branch

stress-ng: info: [19203] skipped: 0

stress-ng: info: [19203] passed: 2: branch (2)

stress-ng: info: [19203] failed: 0

stress-ng: info: [19203] metrics untrustworthy: 0

stress-ng: info: [19203] successful run completed in 10.01 secs

Command being timed: "stress-ng --branch 2 --far-branch-flush --timeout 10s"

User time (seconds): 19.92

System time (seconds): 0.05

Percent of CPU this job got: 198%

Elapsed (wall clock) time (h:mm:ss or m:ss): 0:10.04

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 18552

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 47

Minor (reclaiming a frame) page faults: 3497

Voluntary context switches: 9

Involuntary context switches: 21

Swaps: 0

File system inputs: 0

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

PID 19202 done. real=10.0525s, user=19.9307s, sys=0.054142s

ctx switches: voluntary=11, involuntary=21

exit code: 0

4.2. Тест на 30 минут

Команда для запуска:

mysh$ /usr/bin/time -v stress-ng --branch 10 --far-branch-flush --timeout 2h

Результаты запуска:

mkadilov@mkadilov:~/os/lab1$ /usr/bin/time -v stress-ng --branch 10 --far-branch-flush --timeout 30m

stress-ng: info: [394] setting to a 30 mins run per stressor

stress-ng: info: [394] dispatching hogs: 10 branch

stress-ng: info: [394] skipped: 0

stress-ng: info: [394] passed: 10: branch (10)

stress-ng: info: [394] failed: 0

stress-ng: info: [394] metrics untrustworthy: 0

stress-ng: info: [394] successful run completed in 30 mins

Command being timed: "stress-ng --branch 10 --far-branch-flush --timeout 30m"

User time (seconds): 14389.47

System time (seconds): 6.24

Percent of CPU this job got: 799%

Elapsed (wall clock) time (h:mm:ss or m:ss): 30:00.02

Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0

Average stack size (kbytes): 0

Average total size (kbytes): 0

Maximum resident set size (kbytes): 18416

Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 59

Minor (reclaiming a frame) page faults: 3934

Voluntary context switches: 33

Involuntary context switches: 210628

Swaps: 0

File system inputs: 0

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

Сравнительный анализ теста 4: Branch Predictor (--branch)

Параметр	10 секунд	30 минут	Комментарий
User time	19.92 s	14 389.47 s	User time увеличивается пропорционально числу стрессоров и времени
System time	0.05 s	6.24 s	Системное время минимальное
% CPU	198%	799%	Рост CPU% соответствует увеличению числа стрессоров
Elapsed (real)	10.04 s	1800 s	Реальное время соответствует таймауту
Voluntary context switches	11	33	Незначительный рост

Involuntary context switches	21	210 628	Существенный рост при длительном запуске
Page faults	Minor: 3 497, Major: 47	Minor: 3 934, Major: 59	Незначительное увеличение Minor, Major стабильны

Итоги

Основные наблюдения:

1. **Зависимость времени выполнения и загрузки CPU от числа стрессоров**
    - User time и System time увеличиваются почти пропорционально количеству стрессоров и длительности теста.
    - Например, в тесте **Cache** user time вырос с 10,44 с (2 стрессора, 10 секунд) до 28 758,79 с (10 стрессоров, 2 часа).
    - % CPU также растет с увеличением числа стрессоров, достигая почти полной загрузки процессора в длительных тестах (до ~799%).
2. **Разница в поведении user/system time для разных тестов**
    - В **Instruction Cache** и **Branch Predictor** System time существенно выше при длительном тесте, что связано с особенностями работы с инструкционным кэшем и предсказателем ветвлений.
    - В тестах **Flush Cache** и **Cache** System time остаётся очень низким, user time доминирует.
3. **Переключения контекста (context switches)**
    - Voluntary и involuntary context switches растут в длительных тестах, особенно при большом числе стрессоров.
    - Это показывает активное планирование потоков ОС при полной загрузке CPU.
    - В тесте **Cache** involuntary switches выросли с 482 до 1 334 329, что подтверждает интенсивное использование процессора и кэша.
4. **Page faults (страничные ошибки)**

- Minor page faults существенно увеличиваются в длительных тестах, что связано с активным выделением и освобождением страниц памяти.

- Major page faults остаются относительно стабильными, что говорит о том, что доступ к диску минимален и система эффективно использует оперативную память.

5. **Реальное время выполнения (elapsed / wall clock time)**

    - Реальное время соответствует заданному таймауту, независимо от нагрузки. Это подтверждает корректную работу stress-ng и стабильность системы под нагрузкой.

6. **Сравнение типов стрессоров**

    - **Instruction Cache** и **Branch Predictor** нагружают процессор более интенсивно с точки зрения системного времени, чем тесты **Cache** и **Flush Cache**, что отражает специфику работы CPU с инструкциями.

    - **Cache** и **Flush Cache** генерируют больше minor page faults, что связано с работой с большими объемами данных в кэше.

Вывод

В ходе выполнения лабораторной работы была разработана собственная версия командной оболочки (shell), что позволило глубже понять принципы работы операционной системы с процессами, памятью и контекстом выполнения. В процессе работы я научился запускать и контролировать процессы, работать с системными вызовами, измерять нагрузку на процессор и память, а также анализировать производительность различных подсистем CPU с использованием стресс-тестов. Кроме того, освоение инструментов мониторинга и анализа (stress-ng, /usr/bin/time) позволило оценивать поведение системы под высокой нагрузкой и делать выводы о распределении ресурсов между потоками и процессами.