Отчет по лабораторной работе №5
по курсу "Анализ алгоритмов"
по теме "Конвейерный подход"

Студент: Барсуков Н.М. ИУ7-56
Преподаватель: Волкова Л.Л., Строганов Ю.В.

Содержание

Введение

Конвейеризация или параллелизм? Вот в чем вопрос. Разработчики архитектуры компьютеров издавна прибегали к методам проектирования, известным под общим названием "совмещение операций при котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции. Этот общий метод включает два понятия: параллелизм и конвейеризацию. Хотя у них много общего и их зачастую трудно различать на практике, эти термины отражают два совершенно различных подхода. При параллелизме совмещение операций достигается путем воспроизведения в нескольких копиях аппаратной структуры. Высокая производительность достигается за счет одновременной работы всех элементов структур, осуществляющих решение различных частей задачи.

Конвейеризация (или конвейерная обработка) в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые ступенями, и выделении для каждой из них отдельного блока аппаратуры. Так обработку любой машинной команды можно разделить на несколько этапов (несколько ступеней), организовав передачу данных от одного этапа к следующему. При этом конвейерную обработку можно использовать для совмещения этапов выполнения разных команд. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд. Конвейерная обработка такого рода широко применяется во всех современных быстродействующих процессорах.

1 Аналитический раздел

В данном разделе указана поставленная цель. Описаны задачи необходимые для решения поставленной цели. Рассмотрены 2 подхода выполнения программ: конвейерный и последовательный.

1.1 Постановка задачи

Цель - необходимо разработать 2 версии одной программы, которая обрабатывает строки 2 подходами: последовательным и конвейерным.

Для выполнения данной цели необходимо выполнить следующие задачи:

1) изучить:
   (a) последовательный подход;
   (b) конвейерный подход.
2) выбрать программу для реализации;
3) реализовать выбранную программу двумя подходами, указанными выше;
4) произвести исследования:
   (a) замерить время работы от:
      i. количество слов;
      ii. длины слов
   (b) сравнить;
   (c) сделать выводы.
5) подвести итоги.

1.2 Программа

Наша программа будет работать и выполнять следующие этапы обработки строк:

1) генерировать указанное количество слов в зависимости от входных параметров;
2) приводить их к верхнему регистру;
3) заменять все буквы B на A.

2 Конструкторский раздел

В данном разделе представлена схема основной программы.

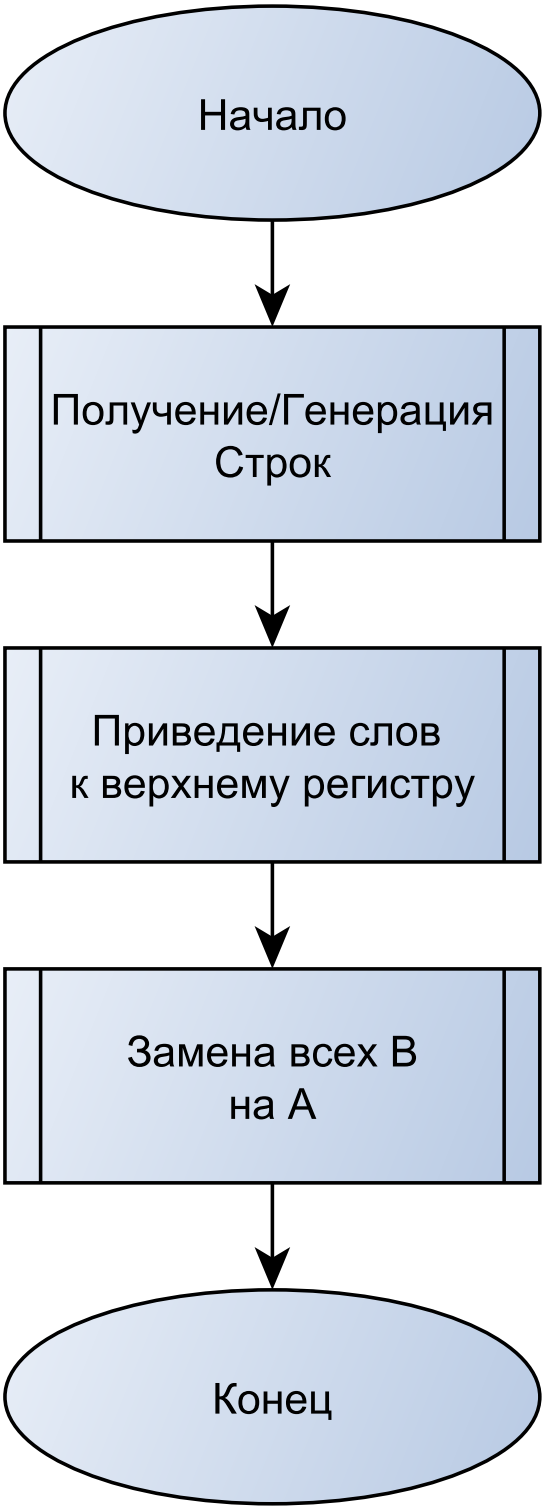

Рис. 1: Схема основной программы

3 Технологический раздел

В данном разделе приведены основные требования к конфигурации машины. Описан используемый язык и среда разработки. Показан интерфейс. Приведен листинг программ.

3.1 Минимальные требования

Минимальные системные требования: PC с операционной системой Windows XP/Vista/7/8/10. Требуются устройства ввода: клавиатура, мышь. Устройство вывода: монитор.

3.2 Выбор языка и среды разработки

Для выполнения поставленной задачи выл выбран язык разработки Erlang/OTP 22.2 с целью изучения новых подходов в программировании. Так же была выбрана Intellij IDEA Education Edition по причине доступности и удобства работы с выбранным языком.

3.3 Интерфейс

Интерфейс программы представляет из себя простую консольную программу которая. Которая просит от пользователя:

1. начальное количество слов;
2. конечное количество слов;
3. шаг изменения
4. длинна слова;
5. количество прогонов

3.4 Листинг

В данном подразделе приведены листинги реализаций двух подходов в выполнении программ: классический последовательный и конвейерный

Листинг 1: Листинг программы в классическом подходе

```
1  -module(simple_way).
2  -author("User").
3
4  %% API
5  -export([test/3, get_random_string/2,
       get_random_strings/3, loop/3]).
6
7
8  test(Words_count, Word_leng, Loops) ->
9  %%  Time_begin = erlang:timestamp(),
```

```
10 %%  loop(Words_count, Word_leng, Loops),
11 %%  timer:now_diff(erlang:timestamp(), Time_begin).
12   timer:tc(fun loop/3, [Words_count, Word_leng, Loops]).
13
14 loop(Words_count, Word_lenght, 0) ->
15   ok;
16
17 loop(Words_count, Word_lenght, Loops) ->
18   Strings = get_random_strings(Words_count, Word_lenght,
         "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM"),
19   Strings1 = lists:map(fun string:to_lower/1, Strings),
20   Func = fun(String) ->
21     string:replace(String, "a", "b", all)
22     end,
23   Strings2 = lists:map(Func, Strings1),
24   loop(Words_count, Word_lenght, Loops - 1).
25
26
27 get_random_strings(0, Length, AllowedChars) ->
28   [];
29 get_random_strings(Count, Length, AllowedChars) ->
30   lists:append([get_random_string(Length,
         AllowedChars)], get_random_strings(Count - 1,
         Length, AllowedChars)).
31
32
33 get_random_string(Length, AllowedChar) ->
34   lists:foldl(fun(_, Acc) ->
35     [lists:nth(rand:uniform(length(AllowedChar)),
           AllowedChar)] ++ Acc end,
36     [], lists:seq(1, Length)).
```

Листинг 2: Листинг программы в конвейерном подходе

```
1 -module(conveerway).
2
3 -export([run_test/3, get_random_strings/3,
      get_random_string/2, controller/4, atob/0,
      tolower/0]).
4
5 run_test(String_amount, Length, Loop_amount) ->
6   Time = erlang:timestamp(),
7   register(controller, spawn(conveer_way, controller,
        [String_amount, Length, Loop_amount, Time])).
8
9 controller(_, _, 0, Time) ->
10   Time1 = erlang:timestamp(),
11   {ok, File} = file:open("out.txt", [append]),
12   io:format(File, "~n~w", [timer:now_diff(Time1, Time)]),
13   io:format("Test done!!! ~n");
```

```
14
15 controller(String_amount, Length, Loop_amount, Time) ->
16   init_test(Length, String_amount,
         "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM"),
17   Monitor1 = erlang:monitor(process, atob),
18   receive
19     {'DOWN', Monitor1, process, Pid, Reason} ->
20 %%       io:format("Loop finished!~n"),
21       controller(String_amount, Length, Loop_amount - 1,
           Time)
22   end.
23
24 init_test(String_length, String_Amount, AllowedChars) ->
25   register(tolower, spawn(conveer_way, tolower, [])),
26   register(atob, spawn(conveer_way, atob, [])),
27   spawn(conveer_way, get_random_strings, [String_Amount,
       String_length, AllowedChars]).
28
29
30 get_random_strings(0, _, _) ->
31   tolower ! finished,
32   exit(normal);
33 get_random_strings(String_Amount, Length, AllowedChars)
     ->
34   String = get_random_string(Length, AllowedChars),
35 %%  io:format("String: ~s ~n", [String]),
36   tolower ! {string, String},
37   get_random_strings(String_Amount - 1, Length,
       AllowedChars).
38
39
40 tolower() ->
41   receive
42     {string, String} ->
43       String_new = string:to_lower(String),
44 %%       io:format("tolower: ~s | ~s ~n", [String,
     String_new]),
45       atob ! {string, String_new},
46       tolower();
47     finished ->
48       atob ! finished,
49       exit(normal)
50   end.
51
52
53 atob() ->
54   receive
55     {string, String} ->
56       string:replace(String, "a", "b", all),
57 %%       io:format("atob: ~s | ~s ~n", [String,
```

```
    String_new]),
58        atob();
59      finished ->
60        exit(normal)
61    end.
62
63
64  get_random_string(Length, AllowedChar) ->
65    lists:foldl(fun(_, Acc) ->
66      [lists:nth(rand:uniform(length(AllowedChar)),
            AllowedChar)] ++ Acc end,
67      [], lists:seq(1, Length)).
```

4 Иследовательский раздел

В данном разделе приведены характеристики оборудования на котором производилось тестирование. Описаны параметры тестирования. Проводятся результаты измерений.

4.1 Характеристики оборудования

1) Компьютер:
   (a) Тип компьютера Компьютер с ACPI на базе x64;
   (b) Операционная система Microsoft Windows 10 Pro.
2) Системная плата:
   (a) тип ЦП DualCore Intel Core i5-6200U, 2700 MHz (27 x 100);
   (b) системная плата HP 8079;
   (c) чипсет системной платы Intel Sunrise Point-LP, Intel Skylake-U;
   (d) системная память 8072 МБ (DDR4 SDRAM).

4.2 Результаты замеров

В данном подразделе приведены результаты замеров в таблицу 4.2. Представлен график зависимости времени от количества слов 2

Замеры проводились по следующим параметрам:

1) Количество итераций: 1000;
2) Количество слов: 10 - 10000;
3) Длинна слов: Фиксированная, 10 символов

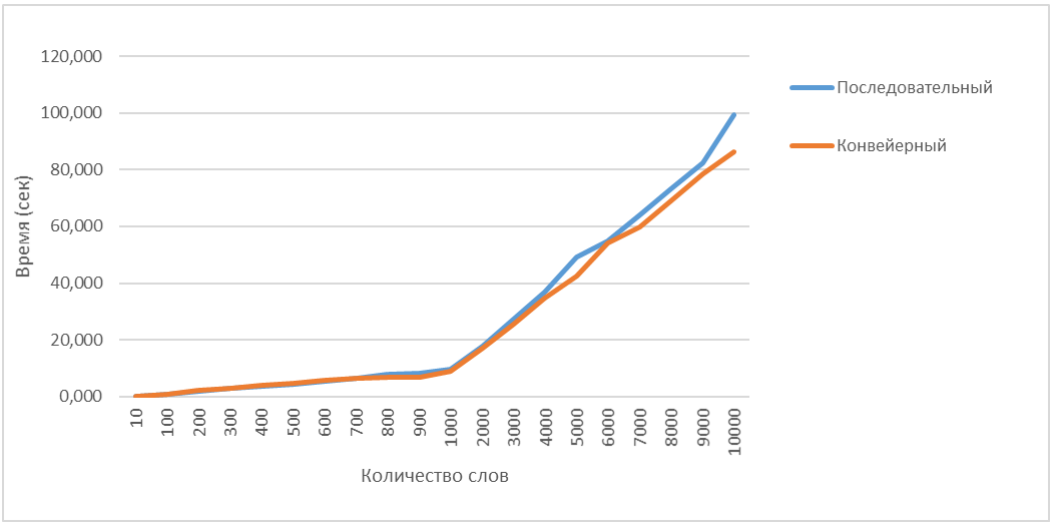

Рис. 2: График зависимости времени от количества слов

	Способ реализации		
Количество слов	Последовательный	Конвеерный	diff
10	0,109	0,110	0,991
100	0,938	0,922	1,017
200	1,766	2,094	0,843
300	3,047	2,813	1,083
400	3,609	3,953	0,913
500	4,328	4,578	0,945
600	5,344	5,703	0,937
700	6,594	6,282	1,050
800	7,782	6,860	1,134
900	8,200	6,860	1,195
1000	9,687	8,829	1,097
2000	17,687	17,125	1,033
3000	27,297	25,531	1,069
4000	37,000	34,578	1,070
5000	49,031	42,469	1,155
6000	54,859	53,984	1,016
7000	64,094	59,750	1,073
8000	73,375	68,859	1,066
9000	82,344	78,672	1,047
10000	99,311	86,344	1,150

Таблица 1: Таблицы измерений

Не смотря на то что вычисляемая задача требовала минимальных вычислительных мощностей, мы можем видеть, что конвейерный подход, на количестве слов большее 2000 начинает показывать производительность лучше, чем классический последовательный метод реализации (хоть и достаточно маленькую в 1.033).

4.3 Вывод

В данном разделе приведены характеристики оборудования на котором проводилось тестирование Указаны результаты замеров эффективности 2 подходов к выполнению задачи.

5 Заключение

В данной лабораторной работе были реализованны два различных подхода в выполнении задач: конвейерный и последовательный. Проведены замеры результатов эффективности методов которые показали, что не смотря на простоту выполняемой задачи преобразования строк, конвейерный метод оказывается эффективнее последовательного, хотя на малых количествах строк они и идут вровень. На пример на количестве строк равном 2000 классический подход работает в 1,033 медленнее конвейерного (на 0.562 секунды медленнее). Но чем больше слов тем больше разница, так для 10000 слов классический работает в 1,150 медленнее чем его коллега. (на 13 секунд медленнее).

Список литературы

[1] Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход.-М.:Техносфера, 2009.

[2] Erlang/OTP 22.2 Электронный ресурс / Режим доступа https://erlang.org/doc/index.html. последнее обращение 25.12.2019

[3] Пензенский Государственный Университет / Конвейеризация и параллелизм. Конвейерная организация обработки данных. Простейшая организация конвейера и оценка его производительности. / Электронный ресурс / Режис доступа https://studfile.net/preview/3991398/page:17. Последнее обращение 25.12.2019