Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ
по практикуму
Задание №1

Тема практикума «Обработка и визуализация графов.»

Название «Разработка и отладка программ в вычислительном комплексе Тераграф»

Дисциплина «Архитектура ЭВМ

Студент: _____ Ву Хай Данг
подпись, дата Фамилия, И.О.

Преподаватель: _____ Ибрагимов С. В.
подпись, дата Фамилия, И. О.

Москва — 2023 г.

Содержание

Цель работы

Практикум посвящен освоению принципов работы вычислительного комплекса Терраграф и получению практических навыков решения задач обработки множеств на основе гетерогенной вычислительной структуры. В ходе практикума необходимо ознакомиться с типовой структурой двух взаимодействующих программ: хост-подсистемы и программного ядра sw_kernel. Для выполнения практикума предоставляется доступ к облачной платформе devlab.bmstu.ru с установленными ускорительными картами микропроцессора Леонард Эйлер и настроенными средствами сборки проектов.

1 Индивидуальное задание

Вариант 20: Устройство формирования индексов SQL EXCEPT. Сформировать в хост-подсистеме и передать в SPE 256 записей множества A (случайные числа в диапазое 0..1024) и 256 записей множества B (случайные числа в диапазоне 0..1024). Сформировать в SPE множество C = A not B. Выполнить тестирование работы SPE, сравнив набор ключей в множестве C с ожидаемым.

2 Экспериментальная часть

2.1 Код программного обеспечения

2.1.1 Host

Листинг 2.1 – Измененный код хост-системы под индивидульное задание

```
1 #include <iostream>
2 #include <iterator>
3 #include <string>
4 #include <regex>
5 #include <sstream>
6 #include <fstream>
7 #include <ctime>
8 #include "host_main.h"
9 using namespace std;
10
11 uint64_t MAX_RECORD = 256;
12 uint64_t MAX = 1024;
13
14 int main(int argc, char** argv)
15 {
16     srand(time(NULL));
17     ofstream log("practicum.log"); //поток вывода сообщений
18     gpc *gpc64_inst; //указатель на класс gpc
19
20     //Инициализация gpc
21     if (argc<2)
22     {
23         log<<"Использование: host_main <путь к файлу
               rawbinary>"<<endl;
24         return -1;
25     }
26
27     //Захват ядра gpc и запись sw_kernel
28     gpc64_inst = new gpc();
29     log<<"Открывается доступ к "<<gpc64_inst->gpc_dev_path<<endl;
30     if (gpc64_inst->load_swk(argv[1])==0) {
```

```
31          log<<"Программное␣ядро␣загружено␣из␣файла␣
                "<<argv[1]<<endl;
32      }
33      else {
34          log<<"Ошибка␣загрузки␣sw_kernel␣файла␣<<␣argv[1]"<<endl;
35          return -1;
36      }
37      //готов данные
38      uint64_t A[MAX_RECORD];
39      uint64_t B[MAX_RECORD];
40      uint64_t C[MAX_RECORD];
41      ifstream fileA("./test/A");
42      ifstream fileB("./test/B");
43      ifstream fileC("./test/C");
44      uint64_t x;
45      int count = 0;
46      while(fileA >> x)
47      A[count++] = x;
48      count = 0;
49      while(fileB >> x)
50      B[count++] = x;
51      count = 0;
52      while(fileC >> x)
53      C[count++] = x;
54      fileA.close();
55      fileB.close();
56      fileC.close();
57      // Инициализация
58      gpc64_inst->start(__event__(update)); //обработчик вставки
59
60      for (uint64_t j = 0; j < MAX_RECORD; j++)
61      {
62          gpc64_inst->mq_send(A[j]);
63      }
64      for (uint64_t j = 0; j < MAX_RECORD; j++)
65      {
66          gpc64_inst->mq_send(B[j]);
67      }
68      //Терминальный символ
69      gpc64_inst->mq_send(-1ull);
70
```

```
71      gpc64_inst->start(__event__(expect_not)); //обработчик запроса
            not
72
73      uint64_t k = gpc64_inst->mq_receive();
74      cout<< k << endl;
75      cout<< gpc64_inst->mq_receive() << endl;
76      cout<< gpc64_inst->mq_receive() << endl;
77
78      count = 0;
79      int err = 0;
80      while (1)
81      {
82          uint64_t key = gpc64_inst->mq_receive();
83          if (key == -1ull) break;
84          if (C[count++] != key)
85          err++;
86      }
87      cout << "Ошибок: " << err <<endl;
88      return 0;
89  }
```

2.1.2 sw_kernel

Листинг 2.2 – Измененный код sw_kernel под индивидульное задание

```
1 #include <stdlib.h>
2 #include <ctime>
3 #include "lnh64.hxx"
4 #include "gpc_io_swk.h"
5 #include "gpc_handlers.h"
6 // #include "iterators.h"
7 // #include "common_struct.h"
8 #include "compose_keys.hxx"
9
10 #define __fast_recall__
11 #define LEFT_STRUCT 1
12 #define RIGHT_STRUCT 2
13 #define RESULT_STRUCT 4
14 #define MAX 256
15
```

```
16 extern lnh lnh_core;
17 volatile unsigned int event_source;
18 uint64_t LNH_key;
19 int main(void) {
20     ////////////////////////////////////////////////////////
21     //                    Main Event Loop
22     ////////////////////////////////////////////////////////
23     //Leonhard driver structure should be initialised
24     lnh_init();
25     for (;;) {
26         //Wait for event
27         event_source = wait_event();
28         switch(event_source) {
29             ////////////////////////////////////////////
30             //  Measure GPN operation frequency
31             ////////////////////////////////////////////
32             case __event__(update) : update(); break;
33             case __event__(expect_not) : expect_not(); break;
34         }
35         set_gpc_state(READY);
36     }
37 }
38
39 //-------------------------------------------------------------------
40 //      Вставка ключа и значения в структуру
41 //-------------------------------------------------------------------
42
43 void update()
44 {
45     lnh_del_str_sync(LEFT_STRUCT);
46     lnh_del_str_sync(RIGHT_STRUCT);
47     int count = 0;
48     while(1)
49     {
50         LNH_key = mq_receive();
51         if (LNH_key==-1ull) break;
52         if (count < MAX)
53         lnh_ins_async(LEFT_STRUCT, LNH_key, LNH_key); //Вставка в
               таблицу с типизацией uint64_t
54         else
55         {
```

```
56              lnh_ins_async(RIGHT_STRUCT, LNH_key, LNH_key);
                    //Вставка в таблицу с типизацией uint64_t
57          }
58          count++;
59      }
60
61  }
62
63  //-------------------------------------------------------------------
64  //      not
65  //-------------------------------------------------------------------
66
67  void expect_not()
68  {
69
70      mq_send(lnh_get_num(LEFT_STRUCT));
71      mq_send(lnh_get_num(RIGHT_STRUCT));
72      lnh_not_sync(LEFT_STRUCT, RIGHT_STRUCT, RESULT_STRUCT);
73      mq_send(lnh_get_num(RESULT_STRUCT));
74
75      uint64_t count = lnh_get_num(RESULT_STRUCT);
76      lnh_get_first(RESULT_STRUCT);
77      for (uint64_t i = 0; i < count; i++)
78      {
79          mq_send(lnh_core.result.key);
80          lnh_next(RESULT_STRUCT, lnh_core.result.key);
81      }
82      mq_send(-1ull);
83
84  }
```

2.2 Тестирование программного обеспечения

Тестирование пройдено успешно.

2.3 Вывод

В ходе практикума было проведено ознакомление с типовой структурой двух взаимодействующих программ: хост-подсистемы и программного ядра sw_kernel. Была разработана программа для хост-подсистемы и обработчика программного ядра, выполняющия действия, описанные в индивидуальном задании