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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О т ч е т

по практикуму № 1

Название: Разработка и отладка программ в вычислительном комплексе Тераграф

Дисциплина: Архитектура ЭВМ

Студент гр. ИУ7-52Б _____ С. С. Беляк
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ А. Ю. Попов
(Подпись, дата) (И.О. Фамилия)

2024 год

Цель работы

Цель работы - освоение принципов работы вычислительного комплекса Терраграф и получение 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

Запуск демо-проекта 1.

```
Open gpc on /dev/gpc2
Rawbinary loaded from sw-kernel/sw_kernel.rawbinary
sw_kernel version: 0x28102023
Leonhard clock frequency (LNH_CF) 180.029670 MHz
Test done
```

Рисунок 1 – Результаты запуска демо-проекта 1

Запуск демо-проекта 2.

```
select role from users where user=5 and time>7200;
Роль: 999 - Время доступа: 3596400
Роль: 998 - Время доступа: 3592800
Роль: 997 - Время доступа: 3589200
Роль: 996 - Время доступа: 3585600
Роль: 995 - Время доступа: 3582000
Роль: 994 - Время доступа: 3578400
Роль: 993 - Время доступа: 3574800
Роль: 992 - Время доступа: 3571200
Роль: 991 - Время доступа: 3567600
Роль: 990 - Время доступа: 3564000
Роль: 989 - Время доступа: 3560400
Роль: 988 - Время доступа: 3556800
Роль: 987 - Время доступа: 3553200
```

Рисунок 2 – Результаты запуска демо-проекта 2

```
Открывается доступ к /dev/gpc2
Программное ядро загружено из файла sw-kernel/sw_kernel.rawbinary
Введен запрос: select role from users where user=5 and time>7200;
Запрос принят в обработку.
Поиск ролей пользователя 5и time > 7200
Введен запрос: exit
Выход!
```

Рисунок 3 – Лог-файл демо-проекта 2

Запуск демо-проекта 3.

```
Утилита для мониторинга состояния гетерогенных ядер обработки графов.
Использование:  lnh_nfo -t/j [-l0,6,..]
-t - вывод в виде таблицы (по умолчанию); -j - вывод в виде json; -l - список ядер gpc (номера через запятую)
Условные обозначения:  <ключи> - количество ключей в структурах 1..7;  <атрибуты> - состояние b+поддеревьев 0..7;
<O> - флаг переполнения поддерева; <E> - флаг пустого поддерева; <S> - номер структуры, занимающей поддерево;
<busy> - ядро выполняет обработчик; <rdy> - ядро ожидает запуска обработчика; * - символическое устройство открыто
```

ядро	параметр	#0	#1	#2	#3	#4	#5	#6	#7
* 0 busy	ключи	-	27000000	19214174	0	0	0	0	0
	атрибуты	O=1 E=0 S=1	O=0 E=1 S=1	O=0 E=0 S=2	O=0 E=0 S=2	O=0 E=0 S=2	O=0 E=0 S=2	O=0 E=0 S=2	O=0 E=0 S=2
* 1 busy	ключи	-	1000000	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=1	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0
2 busy	ключи	-	1000000	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=1	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0
* 3 busy	ключи	-	58720256	5475525	0	0	0	0	0
	атрибуты	O=1 E=0 S=1	O=1 E=0 S=1	O=1 E=0 S=1	O=1 E=0 S=1	O=0 E=1 S=2	O=0 E=1 S=2	O=0 E=1 S=2	O=0 E=1 S=2
6 busy	ключи	-	1000000	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=1	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0
12 rdy	ключи	-	0	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0
18 rdy	ключи	-	0	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0
19 rdy	ключи	-	0	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0
20 rdy	ключи	-	0	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0
21 rdy	ключи	-	0	0	0	0	0	0	0
	атрибуты	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0	O=0 E=1 S=0

Рисунок 4 – Результаты запуска демо-проекта 3

Индивидуальное задание.
Был выполнен вариант 27.

Условие задачи.
Сформировать в хост-подсистеме и передать в SPE две коллекции.

Описание коллекций:

students: student_id, name.

meal_plans: plan_id, student_id, plan_type, start_date, end_date.

Все текстовые поля коллекций предварительно индексируются и сохраняются в std::map в хост-подсистеме (например, путем автоинкремента индекса). В SPE передаются только индексы.

Задание:

определить, есть ли у студента Олег Никитин (передается в запросе из хост-подсистемы) активный план питания на текущую дату (передается в запросе из хост-подсистемы)?

Измененный файл common_struct.h.

```
#ifndef COMMON_STRUCT
#define COMMON_STRUCT

#ifdef __riscv64__
#include "map.h"
#endif
#include "compose_keys.hxx"


//Номера структур данных в SPE
enum Structures : uint32_t {
    null            = 0,        //Нулевая структура не используется
    users_pnum       = 1, //Таблица 1
    resources_pnum  = 2         //Таблица 2
};

#ifdef __riscv64__
//Задание диапазонов и курсоров
template<typename Range>
struct reverse {
        Range r;
        [[gnu::always_inline]] reverse(Range r) : r(r) {}
        [[gnu::always_inline]] auto begin() {return r.rbegin();}
        [[gnu::always_inline]] auto end() {return r.rend();}
};

template<typename K, typename V>
struct Handle {
        bool ret_val;
        K k{get_result_key<K>()};
        V v{get_result_value<V>()};
        [[gnu::always_inline]] Handle(bool ret_val) :
ret_val(ret_val) {
        }
```

```
        [[gnu::always_inline]] operator bool() const {
                return ret_val;
        }

        [[gnu::always_inline]] K key() const {
                return k;
        }

        [[gnu::always_inline]] V value() const {
                return v;
        }
};
#endif


////////////////////////////////////
// Описание формата ключа и значения
////////////////////////////////////


struct students {
     using vertex_t = uint32_t;
    int struct_number;
    constexpr students(int struct_number) :
struct_number(struct_number) {}
     static const uint32_t idx_bits = 32;
     static const uint32_t idx_max = (1ull << idx_bits) - 1;
     static const uint32_t idx_min = idx_max;

     //Запись для формирования ключей (* - наиболее значимые биты
поля)
     STRUCT(key)
     {
         uint32_t    idx       :idx_bits;  //Поле 0
         uint32_t    student_id    :32;    //Поле 1*
     };

     //Запись для формирования значений
     STRUCT(val)
     {
         uint32_t    filler     :32;       //Поле 0
         uint32_t    name    :32;          //Поле 1*
     };
     //Обязательная типизация
     #ifdef __riscv64__
    DEFINE_DEFAULT_KEYVAL(key, val)
     #endif
};

constexpr students STUDENTS(Structures::users_pnum);


struct meal_plans {
     using vertex_t = uint32_t;
    int struct_number;
    constexpr meal_plans(int struct_number) :
struct_number(struct_number) {}
     static const uint32_t idx_bits = 32;
     static const uint32_t idx_max = (1ull << idx_bits) - 1;
     static const uint32_t idx_min = idx_max;

     //Запись для формирования ключей (* - наиболее значимые биты
поля)
     STRUCT(key)
     {
```

```
        uint32_t    idx       :idx_bits;  //Поле 0
        uint32_t    plan_id    :32;       //Поле 1*
    };

    //Запись для формирования значений
    STRUCT(val)
    {
        uint32_t    student_id :32;       //Поле 0*
        uint16_t    plan_type   :16;           //Поле 1*
       time_t      start_date  :8;            //Поле 2*
        time_t      end_date    :8;       //Поле 3*
    };
    //Обязательная типизация
    #ifdef __riscv64__
   DEFINE_DEFAULT_KEYVAL(key, val)
    #endif
};

constexpr meal_plans MEAL_PLANS(Structures::users_pnum);

#endif //COMMON_STRUCT
```

Измененный файл host_main.cpp.

```
#include <iostream>
#include <iterator>
#include <string>
#include <regex>
#include <sstream>
#include <fstream>
#include <ctime>
#include "host_main.h"

using namespace std;

#define TEST_STUDENT_COUNT 5
#define TEST_PLAN_COUNT 5
#define TEST_NAME_COUNT 5

int main(int argc, char** argv)
{
  ofstream log("lab2.log"); //поток вывода сообщений
  unsigned long long offs=0ull;
  gpc *gpc64_inst; //указатель на класс gpc
  regex select_regex_query("select +(.*?) +from +(.*?) +where
+(.*?)=(.*?) +and +(.*?)=(.*);", //запрос
            std::regex_constants::ECMAScript |
std::regex_constants::icase);

  //Инициализация gpc
  if (argc<2) {
    log<<"Использование: host_main <путь к файлу rawbinary>"<<endl;
    return -1;
  }

  //Захват ядра gpc и запись sw_kernel
  gpc64_inst = new gpc();
  log<<"Открывается доступ к "<<gpc64_inst->gpc_dev_path<<endl;
  if (gpc64_inst->load_swk(argv[1])==0) {
    log<<"Программное ядро загружено из файла "<<argv[1]<<endl;
  }
  else {
    log<<"Ошибка загрузки sw_kernel файла << argv[1]"<<endl;
    return -1;
  }
```

```
  //Инициализация таблицы для вложенного запроса
  gpc64_inst->start(event(update)); //обработчик вставки

    // STUDENTS
  //1-й вариант: пересылка коротких сообщений
  for (uint32_t student=0;student<TEST_STUDENT_COUNT;student++) {
    for (uint32_t idx=0;idx<TEST_NAME_COUNT;idx++,offs+=2) {
      gpc64_inst->mq_send(students::key{.idx        = idx,
                                                .student_id =
student});
      gpc64_inst->mq_send(students::val{.filler     = student,
                                                .name       =
student});
    }
  }
  //2-й вариант: блочная передача
  unsigned long long *bufs = (unsigned long
long*)malloc(sizeof(unsigned long
long)*TEST_STUDENT_COUNT*TEST_NAME_COUNT*2);
  for (uint32_t
student=0,offs=0;student<TEST_STUDENT_COUNT;student++) {
    for (uint32_t idx=0;idx<TEST_NAME_COUNT;idx++,offs+=2) {
      bufs[offs]   = students::key{.idx         = idx,
                                           .student_id = student};
      bufs[offs+1] = students::val{.filler      = student,
                                           .name       = student};
    }
  }
  auto send_buf_ths = gpc64_inst->mq_send(sizeof(unsigned long
long)*TEST_STUDENT_COUNT*TEST_NAME_COUNT*2,(char*)bufs);
  send_buf_ths->join();
  free(bufs);


    // MEAL PLANS
  //1-й вариант: пересылка коротких сообщений
  for (uint32_t student=0;student<TEST_STUDENT_COUNT;student++) {
    for (uint32_t plan=0;plan<TEST_PLAN_COUNT;plan++,offs+=2) {
      gpc64_inst->mq_send(meal_plans::key{.idx         = plan,
                          .plan_id    = student});
      gpc64_inst->mq_send(meal_plans::val{.student_id = student,
                          .plan_type  = (uint16_t)plan,
                          .start_date = time_t(0),
                          .end_date   = time_t(0)});
    }
  }

  // 2-й вариант: блочная передача
  unsigned long long *bufm = (unsigned long
long*)malloc(sizeof(unsigned long
long)*TEST_PLAN_COUNT*TEST_STUDENT_COUNT*2); // переименовали bufc в
bufm
  for (uint32_t
student=0,offs=0;student<TEST_STUDENT_COUNT;student++) {
    for (uint32_t plan=0;plan<TEST_PLAN_COUNT;plan++,offs+=2) {
      bufm[offs]   = meal_plans::key{.idx         = plan,
                       .plan_id    = student};
      bufm[offs+1] = meal_plans::val{.student_id = student,
                       .plan_type  = (uint16_t)plan,
                       .start_date = time_t(0),
                       .end_date   = time_t(0)};
    }
  }
  auto send_buf_thm = gpc64_inst->mq_send(sizeof(unsigned long
long)*TEST_PLAN_COUNT*TEST_STUDENT_COUNT*2,(char*)bufm);
```

```
  send_buf_thm->join();
  free(bufm);

  //Терминальный символ
  gpc64_inst->mq_send(-1ull);
  gpc64_inst->start(event(select)); //обработчик запроса поиска
  while(1) {
    string query1;
    //разбор полей запроса
    smatch match_query1;
    getline(cin, query1);
    log<<"Введен запрос: "<<query1<<endl;
    if (!query1.compare("exit")) {
      gpc64_inst->mq_send(-1ull);
      break;
    }

    if (regex_match (query1, match_query1, select_regex_query) &&
      match_query1[3]=="student" &&
      match_query1[5] == "date") {

      log << "Запрос принят в обработку." << endl;
      log << "Поиск для пользователя " << match_query1[4] << " и
даты = " << time_t(stoi(match_query1[6])) << endl;

      int requested_student = stoi(match_query1[4]);
      gpc64_inst->mq_send(requested_student);
      gpc64_inst->mq_send(stoi(match_query1[6]));

      while (1) {
        uint64_t results = gpc64_inst->mq_receive();
        uint64_t resultm = gpc64_inst->mq_receive();
        if (results!=-1ull && resultm!=-1ull) {
          auto student_key = students::key::from_int(results);
          auto meal_plan = meal_plans::val::from_int(resultm);

          if (student_key.student_id == requested_student) {
            cout << "Для студента с именем [#" <<
student_key.student_id << "]";
            cout << " предпочитаемый пищевой план - " <<
meal_plan.plan_type << endl;
            break;
          }
        } else {
          break;
        }
      }
    }else {
          log << "Ошибка в запросе!" << endl;
    }
  }
  log << "Выход!" << endl;
  return 0;
}
```

Измененный файл sw_kernel_main.cpp.

```
#include <stdlib.h>
#include <ctime>
#include "lnh64.hxx"
#include "gpc_io_swk.h"
#include "gpc_handlers.h"
//#include "iterators.h"
#include "common_struct.h"
#include "compose_keys.hxx"

#define __fast_recall__

extern lnh lnh_core;
volatile unsigned int event_source;

int main(void) {
    ////////////////////////////////////////////////////
    //                  Main Event Loop
    ////////////////////////////////////////////////////
    //Leonhard driver structure should be initialised
    lnh_init();
    for (;;) {
        //Wait for event
        event_source = wait_event();
        switch(event_source) {
            ////////////////////////////////////////////
            //  Measure GPN operation frequency
            ////////////////////////////////////////////
            case __event__(update) : update(); break;
            case __event__(select) : select(); break;
        }
        set_gpc_state(READY);
    }
}

//-----------------------------------------------------------------
//      Вставка ключа и значения в структуру
//-----------------------------------------------------------------

void update() {

        while(1){
                students::key
keys=students::key::from_int(mq_receive());
                if (keys==-1ull) break;
                students::val
vals=students::val::from_int(mq_receive());

                STUDENTS.ins_async(keys,vals);

                meal_plans::key
keym=meal_plans::key::from_int(mq_receive());
                if (keym==-1ull) break;
                meal_plans::val
valm=meal_plans::val::from_int(mq_receive());

                MEAL_PLANS.ins_async(keym,valm);
        }
}


//-----------------------------------------------------------------
//      Передать все роли пользователя и время доступа
//-----------------------------------------------------------------
```

```
void select() {
        while(1){
                uint32_t qstudent = mq_receive();
                if (qstudent==-1) break;
                uint32_t qdate = mq_receive();

                auto cname =
STUDENTS.nsm(students::key{.idx=students::idx_min,.student_id=qstude
nt});
                auto cdate =
MEAL_PLANS.nsm(meal_plans::key{.idx=meal_plans::idx_min,.plan_id=mea
l_plans::idx_min});

                while (cname && cname.key().student_id == qstudent)
{
                    if (cdate.value().start_date >= qdate &&
                        cdate.value().end_date <= qdate) {
                       mq_send(cname.value());
                       mq_send(cdate.value());
                    }

                    cname = STUDENTS.nsm(cname.key());
                    cdate = MEAL_PLANS.nsm(cdate.key());
                }

                mq_send(-1ull);
        }
}
```

Результат работы программы.

```
select name from students where student=2 and date=0;
Для студента с именем [#2] предпочитаемый пищевой план - 2

select name from students where student=4 and date=0;
Для студента с именем [#4] предпочитаемый пищевой план - 4
```

Вывод.

В результате работы изучены принципы работы вычислительного комплекса Терапраф и получены практические навыки решения задач обработки множеств на основе гетерогенной вычислительной структуры.