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»

Практикум №1
по дисциплине «Архитектура ЭВМ»

Тема Разработка и отладка программ в вычислительном комплексе Терраграф

Студент Тузов Даниил Александрович

Группа ИУ7-52Б

Преподаватель Калитвенцев Максим Павлович

Москва, 2024 г.

1 Введение

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.

2 Индивидуальное задание

Сформировать в хост-подсистеме и передать в SPE две коллекции. Описание коллекций:

— $students : student_id, name, enrollment_status$.

— $financial_aid : aid_id, student_id, amount, year$.

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

Определить, получал ли студент Артур Назаров (передается в запросе из хост-подсистемы) финансовую помощь в год его зачисления?

Эквивалентный запрос на языке AQL:

Листинг 1 – Запрос на языке AQL

```
FOR student IN students
  FILTER student.name == Артур Назаров
  LET enrollmentYear = DATE_YEAR(student.enrollment_status)
  LET aid = FIRST(
    FOR fa IN financial_aid
      FILTER fa.student_id == student.student_id AND fa.year == enrollmentYear
      RETURN fa
  )
  RETURN {
    name: student.name,
    received_aid_in_enrollment_year: aid != null
  }
```

Объяснение:

1. Находим студента:
   FILTER student.name == Артур Назаров.

2. Получаем год зачисления:
   LET enrollmentYear = DATE_YEAR(student.enrollment_status).

3. Ищем запись о финансовой помощи в год зачисления:

- FILTER fa.student_id == student.student_id
  AND fa.year == enrollmentYear.
- Используем FIRST для получения первой записи.

4. Возвращаем результат:
   - Если aid != null, значит студент получал помощь.
   - RETURN { name: ..., received_aid_in_enrollment_year: ... }.

3 Описание структур

В листинге 2 приведены описания структур: структуры студента и финансовых выплат. Поля структуры соответствуют заданию.

Листинг 2 – Описание структур

```
struct students {
    using vertex_t = uint32_t;
    int struct_number;
    constexpr students(int struct_number) : struct_number(struct_number) {}
    STRUCT(key) {
        uint32_t student_id;
    };
    STRUCT(val) {
        uint32_t name_idx;
        uint32_t enrollment_status;
    };
    #ifdef __riscv64__
        DEFINE_DEFAULT_KEYVAL(key, val)
    #endif
};

struct financial_aids {
    using vertex_t = uint32_t;
    int struct_number;
    constexpr financial_aids(int struct_number) : struct_number(struct_number)
       {}
    STRUCT(key) {
        uint32_t aid_id;
    };
    STRUCT(val) {
        uint16_t student_id;
        uint16_t amount;
        uint16_t year;
    };
    #ifdef __riscv64__
        DEFINE_DEFAULT_KEYVAL(key, val)
    #endif
};
```

4 Описание хост-подсистемы

В листинге 3 приведено описание хост-подсистемы, которая инициализирует *map* студентов, занимает ядро и инициализирует поток сообщений программному ядру с информацией о студентах и финансовых выплатах. Затем формируется запрос от хост-подсистемы к ядру и по результатам этого запроса печатается ответ.

Листинг 3 – Описание хост-подсистемы

```
int main(int argc, char** argv) {
    ofstream log("logger.log"); //поток вывода сообщений
    unsigned long long offs=0ull;
    gpc *gpc64_inst; //указатель на класс gpc
    if (argc<2) {
        log<<"Использование: host_main <путь к файлу rawbinary>"<<endl;
        return -1;
    }
    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;
    }
    log << "Начало инициализации..." << endl;
    gpc64_inst->start(__event__(update_students));
    log << "Таблица студентов" << endl;
    for (const auto& [name, index] : name_index) {
        uint32_t e_status = (1 - index % 2);
        log << "Добавлен студент: id=" << index << " name=" << name << " status
            =" << e_status << endl;
        gpc64_inst->mq_send(students::key{.student_id = index});
        gpc64_inst->mq_send(students::val{.name_idx = index - 1, .
            enrollment_status = e_status});
    }
    gpc64_inst->mq_send(-1ull);
    log << "Инициализирована таблица студентов" << endl;
    gpc64_inst->start(__event__(update_financial_aid));
    log << "Таблица финансовой помощи" << endl;
    for (uint32_t aid_id = 0; aid_id < TEST_AIDS_COUNT; ++aid_id) {
        uint16_t stud_id = aid_id % 10;
```

```
        uint16_t amount = (aid_id % 3) + 1;
        uint16_t year = 2024;
        log << "Добавлена информация о финансовой помощи: id=" << aid_id << "
           stud_id=" << stud_id << " amount=" << amount << " year=" << year <<
           endl;
        gpc64_inst->mq_send(financial_aids::key{.aid_id = aid_id});
        gpc64_inst->mq_send(financial_aids::val{
            .student_id = stud_id,
            .amount = amount,
            .year = year
        });
    }
    gpc64_inst->mq_send(-1ull);
    log << "Инициализирована таблица финансовой помощи" << endl;
    std::string student_name = "Артур Назаров";
    log << "Начало поиска финансовой помощи Артура Назарова" << endl;
    gpc64_inst->start(__event__(select_financials));
    uint32_t name_idx = name_index[student_name];
    gpc64_inst->mq_send(name_idx);
    uint16_t result = gpc64_inst->mq_receive();
    if (result != -1ull) log << "Студент: " << student_name << " получал финанс
       овую помощь в " << result << " году" << endl;
    else log << "Студент: " << student_name << " не получал финансовую помощь"
       << endl;

    delete gpc64_inst;
    return 0;
}
```

5 Описание кода обработчика, функционирующего в ядре

В листинге 4 приведено описание обработчика событий от хост-подсистемы. Обработчик функционирует на базе ядра *sw_kernel*. При получении событий на обновление информации о студентах и выплатах, вызываются функции *update_students* и *update_financial_aid*, которые обновляют информацию о студентах и выплатах. Функция *select_financials* выполняет поиск льгот для переданного в качетсве параметра идентификатора студента. В случае успеха возвращается год соответствующей выплаты.

Листинг 4 – Описание кода обработчика

```
int main(void) {
    lnh_init();
    for (;;) {
        //Wait for event
        event_source = wait_event();
        switch(event_source) {
            case __event__(update_students) : update_students(); break;
            case __event__(update_financial_aid) : update_financial_aid();
               break;
            case __event__(select_financials) : select_financials(); break;
        }
        set_gpc_state(READY);
    }
}

void update_students() {
    while (1) {
        students::key key = students::key::from_int(mq_receive());
        if (key == -1ull) break;
        students::val val = students::val::from_int(mq_receive());
        STUDENTS.ins_async(key, val);
    }
}

void update_financial_aid() {
    while (1) {
        financial_aids::key key = financial_aids::key::from_int(mq_receive());
        if (key == -1ull) break;
        financial_aids::val val = financial_aids::val::from_int(mq_receive());
```

```
        FINANCIAL_AIDS.ins_async(key, val);
    }
}

void select_financials() {
    while (1) {
        uint32_t target_name_idx = mq_receive();
        auto student_iter = STUDENTS.nsm(students::key{.student_id =
           target_name_idx});
        auto aid_iter = FINANCIAL_AIDS.nsm(financial_aids::key{.aid_id = 0});
        while (aid_iter) {
            if (aid_iter.value().student_id == student_iter.key())
                mq_send(aid_iter.value().year);
            aid_iter = FINANCIAL_AIDS.nsm(aid_iter.key());
        }
        mq_send(-1ull);
    }
}
```

6 Вывод

В ходе работы была разработана хост-подсистема, а так же обработчик программного ядра, выполняющие индивидуальное задание. Программа была протестирована — все тесты пройдены успешно.