МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Лабораторные работы №5-7

По курсу «Операционные системы»

Студент: Власко М. М.
Группа: М8О-208Б-23
Преподаватель: Живалев Е. А.
Дата: _____
Оценка: _____
Подпись: _____

Москва, 2024

Тема: Управление серверами сообщений и организация распределённых вычислений

Цель работы: Целью лабораторной работы являлось приобретение практических навыков в:

- управлении серверами сообщений;
- применении отложенных вычислений;
- интеграции программных систем друг с другом.

Вариант: 41 (бинарное дерево поиска, подсчёт суммы чисел, pingall).

Задачи работы:

1. Реализовать распределённую систему асинхронной обработки запросов с использованием технологии очередей сообщений.
2. Создать топологию взаимодействия узлов в виде бинарного дерева поиска.
3. Предусмотреть обработку ошибок и проверку доступности узлов.
4. Реализовать команды:
    - создание нового вычислительного узла;
    - выполнение вычислений на узле;
    - проверка доступности узлов.

Описание решения: Программное решение реализовано на языке C++ с использованием библиотеки ZeroMQ для межпроцессного взаимодействия. Основные модули системы:

1. **Контроллер (Controller):**
    - Принимает команды от пользователя.
    - Создаёт новые вычислительные узлы, добавляя их в бинарное дерево поиска.
    - Отправляет команды узлам и обрабатывает ответы.
    - Реализует асинхронное выполнение команд.
2. **Вычислительные узлы (Worker):**
    - Каждое вычислительный узел создаётся в отдельном процессе с помощью системного вызова `fork()`.
    - Обрабатывают команды на выполнение арифметических операций.
    - Реализуют команду "exec", выполняющую подсчёт суммы заданного количества чисел.

- Отвечают на запросы "ping", подтверждая свою доступность.

3. **Процесс взаимодействия:**
   - Контроллер создаёт процесс узла, передавая ему идентификатор и порт для взаимодействия через ZeroMQ.
   - Команды, такие как "exec" и "ping", передаются через очереди сообщений ZeroMQ в формате строк, а ответы возвращаются обратно в контроллер.
   - Узлы поддерживают механизм связи с другими процессами узлов, что позволяет проверять доступность и взаимодействовать в рамках дерева поиска.
4. **Механизм проверки доступности (Ping):**
   - Рекурсивно проверяет все узлы дерева.
   - Выводит список недоступных узлов.
5. **Обработка ошибок:**
   - Проверка существования узлов, доступности родительских узлов, корректности входных данных.
   - Обработка сбоев связи между узлами и контроллером.

Пример реализации некоторых функций из программы:

```
std::shared_ptr<Node> FindNode(std::shared_ptr<Node> root, int id) {
    if (!root) {
        return nullptr;
    }
    if (root->id == id) {
        return root;
    }
    if (id < root->id) {
        return FindNode(root->left, id);
    }
    return FindNode(root->right, id);
}

bool InsertNode(std::shared_ptr<Node>& root, int id) {
    if (root == nullptr) {
        try {
            root = std::make_shared<Node>(id);
            pid_t pid = fork();
            if (pid == 0) {
                Worker(id, root->sockId);
                exit(0);
            }
```

```
            root->pid = pid;
            return true;
        } catch (zmq::error_t&) {
            return false;
        }
    }
    if (id == root->id) {
        return false;
    }
    if (id < root->id) {
        return InsertNode(root->left, id);
    }
    return InsertNode(root->right, id);
}
```

Пример работы функций: создание нового узла и поиск узла в дереве для последующей команды "exec".

Команды программы: Программа поддерживает следующие команды:

1. **create id [parent]** — создание нового узла с указанным идентификатором.
    - Пример: `create 10 5` -> "Ok: pid"
2. **exec id n k1 ... kn** — выполнение команды подсчёта суммы на указанном узле.
    - Пример: `exec 10 3 1 2 3` -> "Ok:10: 6"
3. **pingall** — проверка доступности всех узлов.
    - Пример: `pingall` -> "Ok: -1" (все узлы доступны).

Репозиторий: https://github.com/MMVlasko/mai-os-lab-work/tree/main/LW567

Исходный код: Программа состоит из следующих файлов:

- `main.cpp`: точка входа, инициализация контроллера.
- `controller.cpp`: управление взаимодействием с пользователем и узлами.
- `worker.cpp`: реализация вычислительных узлов.
- `tools.cpp`: вспомогательные функции для работы с деревом узлов и проверкой доступности.

Пример работы:

```
./lab567

command> create 123

Ok: 76943

command> pingall

Ok: -1

command> exec 123 3 1 2 3

Ok:123: 6

command> exec 124 1 1

Error:124: Not found

command> aaa

Error: Unknown command

command> pingall

Ok: 123

command> exit
```

Вывод: В ходе выполнения работы были достигнуты все поставленные цели. Реализованная распределённая система корректно выполняет задачи асинхронной обработки запросов, поддерживает заданную топологию взаимодействия и обеспечивает устойчивость при сбоях. Программа протестирована в операционной системе Linux и показала стабильную работу. Получены практические навыки работы с библиотекой ZeroMQ, управления процессами и организации межпроцессного взаимодействия.