Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

”Национальный исследовательский университет ИТМО“

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА И РЕАЛИЗАЦИЯ МЕТОДОВ ЭФФЕКТИВНОГО ВЗАИМОДЕЙСТВИЯ ПРОЦЕССОВ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Автор Губарев Владимир Юрьевич _____

Направление подготовки 09.04.04 ”Программная инженерия“

Квалификация Магистр

Руководитель Косяков М.С., к.т.н. _____

Санкт-Петербург, 2020 г.

Обучающийся Губарев В.Ю.
Группа Р42111 Факультет/институт/кластер ПИиКТ

Направленность (профиль), специализация
Информационно-вычислительные системы, Интеллектуальные системы

ВКР принята «_____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «_____» _____ 20___ г.

Секретарь ГЭК Болдырева Е.А. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

”НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО“

УТВЕРЖДАЮ

Руководитель ОП

доцент, д.т.н. Бессмертный И.А. _____

«_____» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Губарев В.Ю.

Группа P42111 **Факультет/институт/кластер** ПИиКТ

Квалификация Магистр

Направление подготовки 09.04.04 ”Программная инженерия“

Направленность (профиль) образовательной программы

Информационно-вычислительные системы

Специализация Интеллектуальные системы

Тема ВКР Разработка и реализация методов эффективного взаимодействия процессов в распределенных системах

Руководитель Косяков М.С., к.т.н., доцент ФПИиКТ

2 Срок сдачи студентом законченной работы «_____» _____ 20___ г.

3 Техническое задание и исходные данные к работе

Требуется разработать и реализовать эффективные методы межпроцессного взаимодействия в пределах одного физического узла. Межпроцессное взаимодействие как с локальными, так и с удаленными процессами должно осуществляться через единый программный интерфейс. Интерфейс должен автоматически выбирать наиболее эффективный метод межпроцессного взаимодействия и скрывать реализацию от пользователя.

4 Содержание выпускной работы (перечень подлежащих разработке вопросов)

а) Обзор предметной области и постановка цели работы.

б) Разработка и реализация методов эффективного взаимодействия процессов.

в) Экспериментальное исследование и обработка результатов.

5 Перечень графического материала (с указанием обязательного материала)

а) Гистограммы временной задержки на передачу данных для разработанных методов межпроцессного взаимодействия.

б) Принципиальные схемы разработанных методов межпроцессного взаимодействия.

6 Исходные материалы и пособия

а) Косяков М.С. Введение в распределенные вычисления. Учебное пособие / М.С. Косяков. – СПб: СПбГУ ИТМО, 2014. – 155 с.

б) Schmidt D.C. et al. Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects. – John Wiley & Sons, 2013. – Т. 2.

7 Дата выдачи задания «_____» _____ 20___ г.

Руководитель ВКР _____

Задание принял к исполнению _____«_____» _____ 20___ г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

”НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО“

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся Губарев Владимир Юрьевич

Наименование темы ВКР: Разработка и реализация методов эффективного взаимодействия процессов в распределенных системах

Наименование организации, в которой выполнена ВКР Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: уменьшение временной задержки на передачу данных между процессами в пределах одного физического узла путем разработки и применения методов эффективного межпроцессного взаимодействия.

2 Задачи, решаемые в ВКР:

а) рассмотреть существующие методы межпроцессного взаимодействия, доступные при взаимодействии процессов, находящихся на одном физическом узле;

б) произвести анализ и отбор методов межпроцессного взаимодействия для реализации новых методов межпроцессного взаимодействия;

в) разработать и реализовать эффективные методы межпроцессного взаимодействия;

г) экспериментально исследовать полученные методы межпроцессного взаимодействия.

3 Число источников, использованных при составлении обзора: 0

4 Полное число источников, использованных в работе: 0

5 В том числе источников по годам:

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	0	0	0

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	**Раздел работы**
LaTeX	Весь текст диссертации и сопроводительные документы
C++17 (“International Standard ISO/IEC 14882:2014(E) Programming Language C++”)	Раздел **??**, Приложение **??**
LTTng	Глава **??**

8 Краткая характеристика полученных результатов

Разработано семейство новых методов межпроцессного взаимодействия в пределах одного физического узла, показавших меньшую временную задержку на передачу данных, чем разработанные ранее.

9 Гранты, полученные при выполнении работы

Отсутствуют.

10 Наличие публикаций и выступлений на конференциях по теме работы

1 *Губарев В. Ю.* Реализация методов эффективного взаимодействия процессов в распределенных системах // Сборник тезисов докладов конгресса молодых ученых. Электронное издание. — Университет ИТМО, 2020.

Обучающийся Губарев В.Ю. _____

Руководитель Косяков М.С. _____

«____» _____ 20__ г.

ABSTRACT

Author: Gubarev Vladimir Yurievich

Development and implementation of efficient inter-process communication methods for distributed systems

Nowadays distributed systems are widely spreaded. They are usually designed to work in various environments as a set of cooperating processes. At the same time capabilities of modern hardware allow to deploy groups of that processes within a single machine in order to achieve better performance. In this case efficient inter-process communication (IPC) methods become a crucial element of high-performance distributed systems.

The present work is focused on developing efficient IPC methods. Based on the most efficient IPC in Linux, shared memory and futex, it introduces new methods of low-latency IPC. They are transparently provided via a generic interface. The interface automatically and transparently for programmer uses TCP to communicate over network with remote processes and low-latency shared memory-based method for local processes.

Proposed methods show significantly lower latency with local processes than TCP-based without any additional difficulties for programmer.

ОГЛАВЛЕНИЕ

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЦЕЛИ РАБОТЫ

1.1. Методы межпроцессного взаимодействия

Метод межпроцессного взаимодействия – это способ осуществления взаимодействия процессов, находящихся на одном или разных физических узлах.

В Linux поддержаны следующие методы межпроцессного взаимодействия:

а) Интернет-сокеты и сокеты домена Unix.
б) именованные и неименованные каналы.
в) Очередь сообщений.
г) Разделяемая память: SystemV и POSIX.

Рассмотрим приведенные выше методы межпроцессного взаимодействия.

1.1.1. Сокеты

Сокет – это программный интерфейс для обеспечения обмена данными между процессами. В зависимости от реализации интерфейса, позволяет взаимодействовать процессам как на разных физических узлах в составе сети, так и в пределах одного узла. Широко распространены два вида, сокеты TCP и сокеты домена Unix. Первый работает через протокол TCP, второй – использует некоторую внутреннюю реализацию канала связи, как и в TCP.

1.1.1.1. TCP-сокеты

TCP-сокет – интерфейс межпроцессного взаимодействия, использующий протокол TCP. Среди всех рассматриваемых методов только он позволяет взаимодействовать процессам на разных физических узлах в составе сети. В то же время, возможно межпроцессное взаимодействие и в пределах одного узла через механизм обратной петли, когда TCP-сообщение передается без каких-либо излишних операций в процесс-получатель, не покидая физического узла. В качестве точки соединения используется пара IP-адрес и порт.

TCP широко распространен. Сторонние системы чаще всего предоставляют именно TCP-интерфейс доступа к своим службам.

1.1.1.2. Сокеты домена Unix

Unix-сокет или IPC-сокет – интерфейс межпроцессного взаимодействия в пределах одного узла, не использующий сетевого протокола. Может работать в разных режимах: передачи потока байт, датаграмм или последовательных пакетов. Первые два соответствуют протоколам TCP и UDP. Третий – последовательный надежный канал для передачи датаграмм.

1.1.2. Каналы

1.1.2.1. Неименованные каналы

Неименованный канал – однонаправленный метод межпроцессного взаимодействия для родственных процессов. Данные, записанные в неименованный канал, остаются там до момента считывания, либо до момента завершения ссылающихся на него процессов. Размер буфера ограничен.

1.1.2.2. Именованные каналы

Именованный канал отличается от неименованного тем, что представлен в виде файла, следовательно, доступен всем процессам ОС. Кроме того, время его жизни не ограничено временем жизни использующих его процессов.

1.1.3. Очередь сообщений

Очередь сообщений – список сообщений, который хранится в пространстве ядра. Каждая очередь имеет свой уникальный идентификатор. Взаимодействие происходит путем вызовов записи и чтения сообщений.

1.1.4. Разделяемая память

Обычно в целях безопасности адресное пространство процесса изолировано от других. В некоторых случаях, однако, может быть необходимо использовать совместно один и тот же сегмент памяти. Такая память называется разделяемой. На рисунке 1 приведен пример использования разделяемой памяти двумя процессами.

Существуют два интерфейса для доступа к разделяемой памяти. Более старый System V и более новый POSIX.

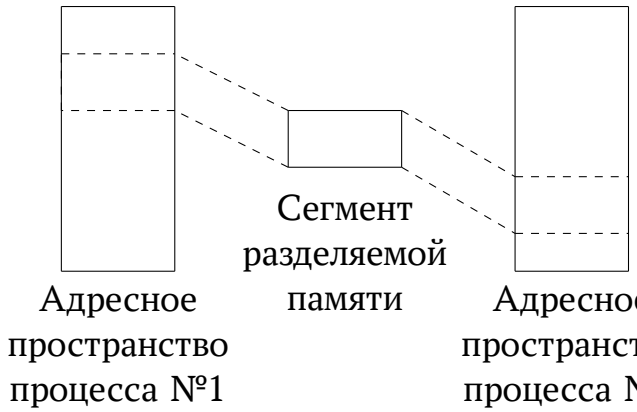

Рисунок 1 – Пример использования разделяемой памяти

1.1.4.1. System V

Разделяемая память здесь является системным ресурсом, она представлена уникальным ключом в пределах ОС.

Интерфейс реализуется через набор системных вызовов и структур для работы с ним:

- shmget(key_t key, size_t size, int oflag) – вызов для создания нового сегмента разделяемой памяти или использования существующего с ключом *key* . Размер сегмента *size*, флагb доступа и создания *oflag*;
- shmat(int shmid, const void *shmaddr, int flag) – подключение сегмента в адресное пространство процесса;
- shmdt(const void *shmaddr) – отключение сегмента от адресного пространства;
- shmctl(int shmid, int cmd, struct shmid_ds *buf) – управление разделяемой памятью: изменение прав доступа, удаление, запрос статистики;

1.1.4.2. POSIX

Разделяемая память здесь является пользовательским ресурсом, она представлена файлом. Интерфейс реализуется через набор системных вызовов:

- shm_open(const char *name, int oflag, mode_t mode) – открывает файл для разделяемой памяти (аналог shmget). В отличие от обычного вызова open, открытый таким образом файл не синхронизируется с диском.
- shm_unlink(const char *name) – удаляет файл (аналог shmctl);
- ftruncate(int fd, off_t length) – задает размер файла;
- fstat(int fd, struct stat *statbuf) – статистика о файле (аналог shmctl);

— mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset) – отображение файла в памяти (аналог shmat);
— munmap(void *addr, size_t length) – отключает отображенный сегмент от адресного пространства процесса (аналог shmdt).

1.1.4.3. Разница между System V и POSIX

Оба интерфейса предлагают аналогичные возможности. Разница состоит в предоставлении ресурса в системе, имени файла для POSIX или целом числе в System V. Также, в POSIX интерфейсе сегмент разделяемой памяти будет уничтожен, когда будет завершен последний процесс, отображающий его в память, а файл полностью удален. Это свойство полезно на случай непредвиденного завершения процессов, например, в результате программного дефекта.

1.1.5. Сравнение методов межпроцессного взаимодействия

Приведенные выше методы можно разделить на группы по трем критериям:
— способности к взаимодействию с удаленными процессами, то есть с процессами на других физических узлах;
— использованию ядра ОС для осуществления межпроцессного взаимодействия.

Среди всех представленных методов только TCP-сокеты позволяют взаимодействовать как процессам на разных физических узлах, так и на одном узле, используя один и тот же интерфейс. Поэтому разумно использовать именно этот метод в качестве базового метода межпроцессного взаимодействия.

Все приведенные методы, за исключением разделяемой памяти, при межпроцессном взаимодействии используют ядро операционной системы. Использование ядра существенно увеличивает временную задержку на передачу данных. Как минимум, из-за двойного копирования данных, перехода между пользовательским режимом и режимом ядра (см. рисунок 2). Разделяемая память же позволяет осуществлять взаимодействие напрямую через пользовательское пространство.

Однако, сама по себе разделяемая не предоставляет возможностей для синхронизации процессов, не имеет механизма оповещения об изменении состояния разделяемой памяти. Для этого необходимо рассмотреть методы синхронизации процессов.

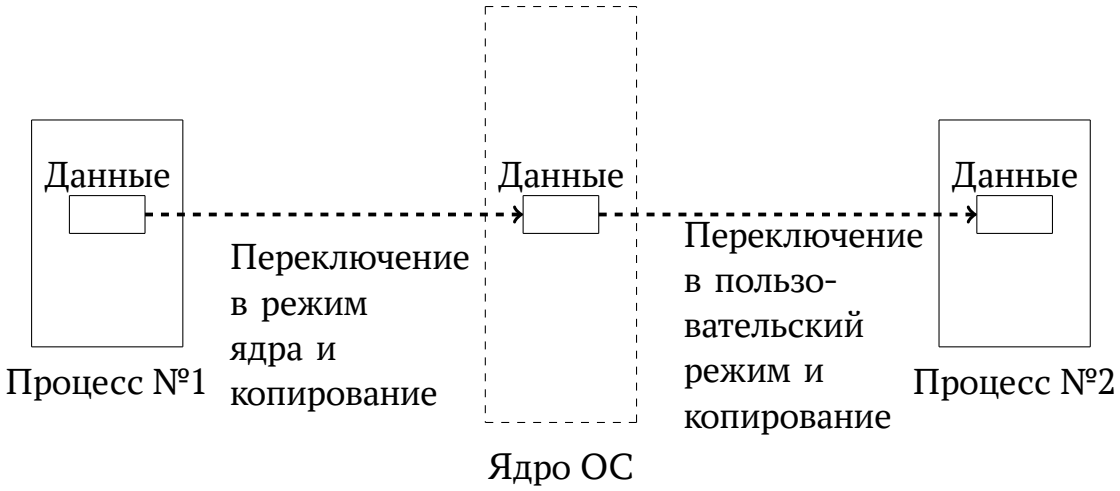

Рисунок 2 – Общая схема межпроцессного взаимодействия при использовании ядра ОС

1.2. Методы синхронизации процессов

Методы синхронизации процессов нужны, чтобы корректно передавать данные между процессами через разделяемую память, например, не допускать состояния гонки. Можно выделить два класса методов: примитивы синхронизации и атомарные операции. Примитивы синхронизации зачастую используют в своих алгоритмах атомарные операции.

1.2.1. Примитивы синхронизации

В Linux на уровне ядра ОС поддержаны следующие примитивы межпроцессной синхронизации:

а) System V семафоры;
б) futex – **f**ast **u**serspace mu**tex**.

В свою очередь, futex служит основой для более продвинутых примитивов синхронизации:

а) POSIX-семафора;
б) mutex – взаимного исключения;
в) rw-mutex – взаимного исключения для писателя и множества читателей.

Рассмотрим их подробнее.

1.2.1.1. futex

1.2.1.2. Семафоры

POSIX

System V

1.2.1.3. Взаимные исключения

Mutex

RW-Mutex

1.2.2. Атомарные операции

TBD: это такие операции..

Виды атомарных операций:

а) load/store – атомарные чтение или запись;

б) swap – атомарно устанавливает значение переменной и возвращает старое значение;

в) compare-and-swap – атомарное изменение значения переменной при истинном значении предиката над переменной; **TBD: подкорректировать**

г) fetch-and-add – атомарная арифметическая или логическая операция над переменной с возвращением предыдущего значнеия;

1.3. Предыдущая работа

TBD: написать про мою бакалаврскую ВКР и сослаться на электронный тезис?

1.4. Обзор литературы

TBD: привести аналогичные исследования в этой области. Чужие диссеры про IPC между процессами, между виртуалками. Много раз сказать, что быстрее shmem ничего нет.

1.5. Коммерческие решения

TBD: informatica и tibco

1.6. Значимость методов на основе разделяемой памяти в межпроцессном взаимодействия

TBD: Таким образом, методы на основе разделяемой памяти...

1.7. Необходимость в едином интерфейса доступа к методам межпроцессного взаимодействия

TBD: Чтобы методами можно было пользоваться, нужен интерфейс, интерфейс был сделан, поэтому теперь все работает через него, включая и эту работу

1.8. Критерий эффективности и постановка цели работы

TBD: временная задержка на передачу данных TBD: и ее цель – ее уменьшение в сравнении с TCP

Выводы по главе 1

TBD: выводы