

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПРИМЕР ОФОРМЛЕНИЯ ВКР МАГИСТРА

Автор: Буздалов Максим Викторович _____

Направление подготовки: 01.04.02 Прикладная
математика и информатика

Квалификация: Магистр

Руководитель: Шалыто А.А., проф., д.т.н. _____

К защите допустить

Руководитель ОП Парфенов В.Г., проф., д.т.н. _____

« ____ » _____ 20 ____ г.

Санкт-Петербург, 2019 г.

Студент Буздалов М.В.

Группа М4239 Факультет ИТиП

Направленность (профиль), специализация

Технологии разработки программного обеспечения

Консультанты:

а) Белашенков Н.Р., канд. физ.-мат. наук, без звания

б) Беззубик В.В., без степени, без звания

ВКР принята « _____ » _____ 20__ г.

Оригинальность ВКР _____ %

ВКР выполнена с оценкой _____

Дата защиты « _____ » _____ 20__ г.

Секретарь ГЭК Павлова О.Н.

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
ГЛАВА 1. Обзор методов межпроцессного взаимодействия	8
1.1. Таблицы.....	8
1.2. Рисунки.....	8
1.3. Листинги.....	8
ГЛАВА 2. Разработка и реализация методов эффективного взаимодействия процессов	11
Выводы по главе 2.....	11
ГЛАВА 3. Анализ результатов.....	12
ЗАКЛЮЧЕНИЕ	13
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	14
ПРИЛОЖЕНИЕ А. Пример приложения	15
ПРИЛОЖЕНИЕ Б. Еще один пример приложения с невероятно длинным названием для тестирования переносов.	17
ПРИЛОЖЕНИЕ В. Пример огромного листинга	18

ВВЕДЕНИЕ

Объектом исследования являются методы межпроцессного взаимодействия.

Предметом исследования является временная задержка на передачу данных между процессами распределенной системы в пределах одного физического узла.

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

В рамках настоящей работы поставлены следующие **задачи**:

- рассмотреть существующие методы межпроцессного взаимодействия, доступные при взаимодействии процессов, находящихся на одном физическом узле;
- произвести анализ и отбор методов межпроцессного взаимодействия для реализации новых методов межпроцессного взаимодействия;
- разработать и реализовать эффективные методы межпроцессного взаимодействия;
- экспериментально оценить полученные методы межпроцессного взаимодействия.

Актуальность исследования.

Для некоторых систем эффективное межпроцессное взаимодействие является критически важной частью их работы. Требование по минимизации времени обслуживания заявок может напрямую следовать из области применения системы, как в случае с системами для алгоритмической торговли на финансовых рынках. Обслуживание заявок множеством логически связанных процессов может быть существенно ускорено при размещении таких процессов на одном физическом узле. Современные процессоры с количеством ядер более **40** обеспечивают такую возможность. Это позволяет использовать более эффективные методы межпроцессного взаимодействия, а именно методы на основе разделяемой памяти [1]. Эффективные методы межпроцессного взаимодействия могут использоваться для связи виртуальных машин или контейнеров в пределах машины-хозяина [4, 5]. Для связи программных модулей, исполняющихся в разных процессах для обеспечения отказоустойчивости за

счет изоляции процессов на уровне ОС. Для высокопроизводительных вычислений, таких как анализ научных данных или прогнозирование погоды.

При разработке сложной многокомпонентной распределенной системы программисту необходимо сосредоточиться на логике и корректности работы самой системы. В то время как методы межпроцессного взаимодействия должны быть для него прозрачны. Этого можно достичь, используя единый унифицированный интерфейс для межпроцессного взаимодействия. Это упрощает разработку, снимает необходимость сложного управления ресурсами для межпроцессного взаимодействия. А также позволяет автоматически использовать наиболее подходящие методы межпроцессного взаимодействия для данных *пространственных конфигураций* (**ТВД: может, убрать?**) процессов, что может повысить эффективность выполнения некоторых задач этой системой.

Таким образом, разработка и реализация эффективных методов межпроцессного взаимодействия и интерфейса для автоматического доступа к наиболее подходящим из них необходима и обоснована. Посредством этого интерфейса программист прозрачно для себя использует методы межпроцессного взаимодействия на основе разделяемой памяти при взаимодействии с локальными процессами без необходимости перекомпиляции программы. Но поскольку зачастую нельзя разместить всю систему на одном, даже очень производительном, сервере используется ТСР при взаимодействии с процессами на других физических узлах.

Методы исследования включают в себя анализ существующих методов межпроцессного взаимодействия, экспериментальную оценку разработанных методов эффективного межпроцессного взаимодействия и методы математической статистики для обработки экспериментальных данных.

Средства исследования:

- язык программирования C++, компилятор *Clang 6.0.1*, стандартная библиотека C++ *libstdc++*;
- Библиотека Boost.Interprocess [2] для управления разделяемой памятью;
- система трассировки событий [6] на основе инструмента LTTng [3].

Научная новизна заключается в предложенных новых методах эффективного межпроцессного взаимодействия в пределах одного физического узла, которые не описаны в существующих исследованиях.

Положения, выносимые на защиту

TBD.

- а) shmem + mux + futex + lf;
- б) shmem + mux + futex + hsha;
- в) shmem + mux + busy wait.

Апробация результатов.

Основные результаты работы были представлены на IX Конгрессе Молодых Ученых.

Предложенные методы межпроцессного взаимодействия на основе разделяемой памяти подготовлены к использованию в платформе для торговли на финансовых рынках Tbricks от компании Itiviti.

ГЛАВА 1. ОБЗОР МЕТОДОВ МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ

Пример ссылок в рамках обзора: [**example-english**, **example-russian**, **unrestricted-jump-evco**, **nsga-ii-steady-state**]. Вне обзора: [**bellman**].

1.1. Таблицы

В качестве примера таблицы приведена таблица 1.

Таблица 1 – Таблица умножения (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Есть еще такое окружение `tabularx`, его можно аккуратно растянуть на всю страницу. Приведем пример (таблица 2).

Таблица 2 – Таблица умножения с помощью `tabularx` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

1.2. Рисунки

Пример рисунка (с помощью `TikZ`) приведен на рисунке 1. Под `pdflatex` можно также использовать `*.jpg`, `*.png` и даже `*.pdf`, под `latex` можно использовать `Metapost`. Последний можно использовать и под `pdflatex`, для чего в стилевике продекларированы номера картинок от 1 до 20.

1.3. Листинги

В работах студентов кафедры «Компьютерные технологии» часто встречаются листинги. Листинги бывают двух основных видов — исходный код и

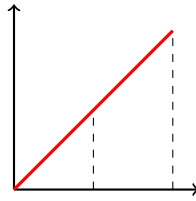


Рисунок 1 – Пример рисунка

Листинг 1 – Пример исходного кода на Java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

псевдокод. Первый оформляется с помощью окружения `lstlisting` из пакета `listings`, который уже включается в стилевике и немного настроен. Пример Hello World на Java приведен на листинге 1. Пример большого листинга — в приложении (листинг B.1).

Псевдокод можно оформлять с помощью разных пакетов. В данном стилевике включается пакет `algorithmicx`. Сам по себе он не генерирует флюатов, поэтому для них используется пакет `algorithm`. Пример их совместного использования приведен на листинге 2.

Листинг 2 – Пример псевдокода

```
function IsPrime( $N$ )
  for  $t \leftarrow [2; \lfloor \sqrt{N} \rfloor]$  do
    if  $N \bmod t = 0$  then
      return false
    end if
  end for
  return true
end function
```

Наконец, листинги из `listings` тоже можно подвешивать с помощью `algorithm`, пример на листинге 3.

Листинг 3 – Исходный код и флот `algorithm`

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
    }  
}
```

ГЛАВА 2. РАЗРАБОТКА И РЕАЛИЗАЦИЯ МЕТОДОВ ЭФФЕКТИВНОГО ВЗАИМОДЕЙСТВИЯ ПРОЦЕССОВ

Листинг 4 должен иметь номер 4.

Листинг 4 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Рисунок 2 должен иметь номер 2.

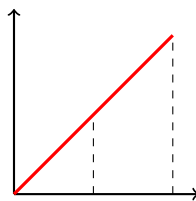


Рисунок 2 – Пример рисунка

Таблица 3 должна иметь номер 3.

Таблица 3 – Таблица умножения с помощью `tabularx` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Выводы по главе 2

В конце каждой главы желательно делать выводы. Вывод по данной главе — нумерация работает корректно, ура!

ГЛАВА 3. АНАЛИЗ РЕЗУЛЬТАТОВ

ЗАКЛЮЧЕНИЕ

В данном разделе размещается заключение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Draft: Have you checked your IPC performance lately? [Электронный ресурс] / S. Smith [и др.]. — 2012. — URL: <https://www.semanticscholar.org/paper/Draft-%3A-Have-you-checked-your-IPC-performance-Smith-Madhavapeddy/1cb3b82e2ef6a95b576573b8af0f3ec6f7bc21d9> (дата обр. 25.03.2020).
- 2 *Gaztanaga I.* Chapter 18. Boost.Interprocess [Электронный ресурс]. — URL: https://www.boost.org/doc/libs/1_63_0/doc/html/interprocess.html (дата обр. 25.03.2020).
- 3 LTTng: an open source tracing framework for Linux [Электронный ресурс]. — URL: <https://lttng.org/> (дата обр. 25.03.2020).
- 4 *Macdonell A. C.* Shared-Memory Optimizations for Virtual Machines : PhD dissertation / Macdonell A. Cameron. — Department of Computing Science : University of Alberta, 2011. — DOI: <https://doi.org/10.7939/R3Q715>.
- 5 *Xiaodi K.* Interprocess Communication Mechanisms With Inter-Virtual Machine Shared Memory : Master's dissertation / Xiaodi Ke. — Department of Computing Science : University of Alberta, 2011. — DOI: <https://doi.org/10.7939/R3PH6H>.
- 6 *Кузичкина А.* Исследование и разработка методов трассировки событий в параллельных и распределенных системах : Пояснительная записка выпускной квалификационной работы / Кузичкина Анастасия. — Факультет ПИиКТ : Университет ИТМО, 06.2017.

ПРИЛОЖЕНИЕ А. ПРИМЕР ПРИЛОЖЕНИЯ

В приложениях рисунки, таблицы и другие подобные элементы нумеруются по приложениям с соответствующим префиксом. Проверим это.

Листинг А.1 должен иметь номер А.1.

Листинг А.1 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Рисунок А.1 должен иметь номер А.1.

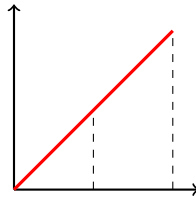


Рисунок А.1 – Пример рисунка

Таблица А.1 должна иметь номер А.1.

Таблица А.1 – Таблица умножения с помощью `tabularx` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Заодно проверим нумерованные и ненумерованные перечисления. Ненумерованные:

- пункт А;
- пункт Б;
- пункт В.

Нумерованные списки нескольких уровней:

- а) первый элемент;
- б) второй элемент с подэлементами:
 - 1) первый подэлемент;

2) второй подэлемент;

3) третий подэлемент.

в) третий элемент;

г) четвертый элемент;

д) пятый элемент;

е) шестой элемент;

ж) седьмой элемент;

и) восьмой элемент;

к) девятый элемент;

л) десятый элемент.

**ПРИЛОЖЕНИЕ Б. ЕЩЕ ОДИН ПРИМЕР ПРИЛОЖЕНИЯ С
НЕИМОВЕРНО ДЛИННЮЩИМ НАЗВАНИЕМ ДЛЯ
ТЕСТИРОВАНИЯ ПЕРЕНОСОВ**

Проверим на примере таблиц, что нумерация в приложениях — по приложениям. Таблица Б.1 должна иметь номер Б.1.

Таблица Б.1 – Таблица умножения с помощью `tabularx` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

ПРИЛОЖЕНИЕ В. ПРИМЕР ОГРОМНОГО ЛИСТИНГА

Листинг В.1 – Пример большого листинга

```
import java.util.*;

public class Example {
    static int[] restoreOutgoing(int[] g, int[] outgoing,
                                int vertex, int mask) {
        int[] rv = new int[1 + Integer.bitCount(mask)];
        int n = g.length;
        int current = rv.length - 1;
        while (true) {
            rv[current] = vertex;
            if (current == 0) {
                if (vertex != 0) {
                    throw new AssertionError();
                }
                return rv;
            }
            mask ^= 1 << (vertex - 1);
            int prevMask = outgoing[mask] & g[vertex];
            if (prevMask == 0) {
                throw new AssertionError();
            }
            vertex = Integer.numberOfTrailingZeros(prevMask);
            --current;
        }
    }

    static int[] restoreIncoming(int[] g, int[] incoming,
                                int vertex, int mask) {
        int[] rv = new int[1 + Integer.bitCount(mask)];
        int n = g.length;
        int current = 0;
        while (true) {
            rv[current] = vertex;
            if (current == rv.length - 1) {
                if (vertex != 0) {
                    throw new AssertionError();
                }
                return rv;
            }
            mask ^= 1 << (vertex - 1);
            int prevMask = incoming[mask] & g[vertex];
            if (prevMask == 0) {
                throw new AssertionError();
            }
            vertex = Integer.numberOfTrailingZeros(prevMask);
            ++current;
        }
    }
}
```

```
mask ^= 1 << (vertex - 1);
int nextMask = incoming[mask] & g[vertex];
if (nextMask == 0) {
    throw new AssertionError();
}
vertex = Integer.numberOfTrailingZeros(nextMask);
++current;
}
}
}
```