

Работу выполнил:
Гайдай А. В.
Научный руководитель:
Кулагин И. И.

Алгоритм оптимизации использования мьютексов по результатам предварительного профилирования

Докладчик: Гайдай А. В.



Эл. почта: diligent20494@gmail.com
Телефон: +7 (983)-309-84-33

Актуальность

- При разработке параллельных программ для обеспечения их корректной работы необходимо избегать возникновения ситуации гонки за данными (data race). Для этой цели используются механизмы взаимного исключения – мьютексы
- Промышленным стандартом реализации мьютексов при разработке параллельных программ является динамическая библиотека pthread в GNU libc
- Реализация мьютексов в текущей версии (glibc 2.23) не учитывает динамически изменяющиеся характеристики критических секций

Мьютексы

- Мьютекс – примитив синхронизации, позволяющий создавать в коде программы критические секции, выполнение которых возможно только одним потоком в каждый момент времени
- Мьютексы могут находиться в одном из двух состояний – открытом или закрытом

```
// Пользовательская хэш-таблица
hash_t  hash;

pthread_mutex_t  mut;

// Мьютекс закрыт
pthread_mutex_lock (&mut);

// Начало критической секции
***

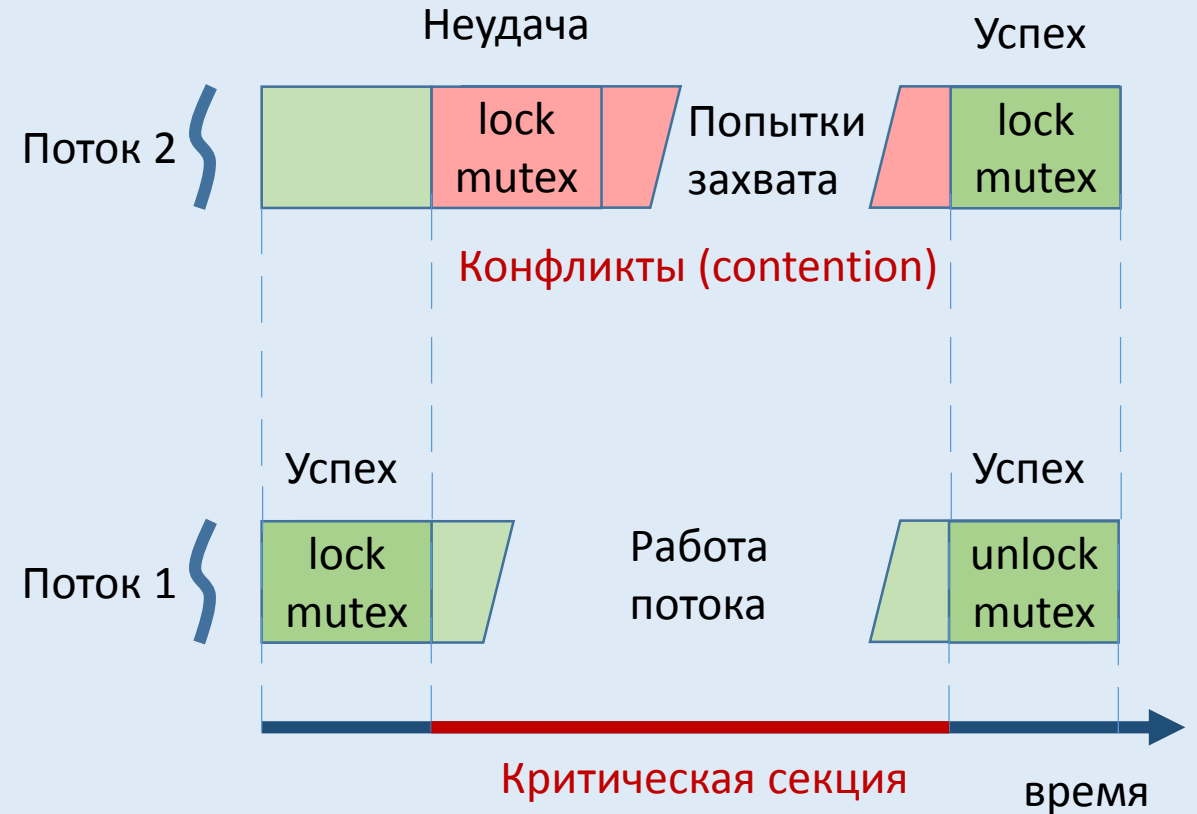
// Изменение хэш-таблицы
hash_t_add (&hash, value);
***

// Конец критической секции

// Мьютекс закрыт
pthread_mutex_unlock (&mut);
```

Конфликты

- В многопоточных программах высока вероятность возникновения ситуации, при которой один поток пытается захватить мьютекс, закрытый другим потоком
- Такая ситуация называется конфликтом (contention)



Конфликты

- Возникающие конфликты негативно сказываются на времени выполнения многопоточных программ в силу особенностей работы протокола когерентности кэшей MESI



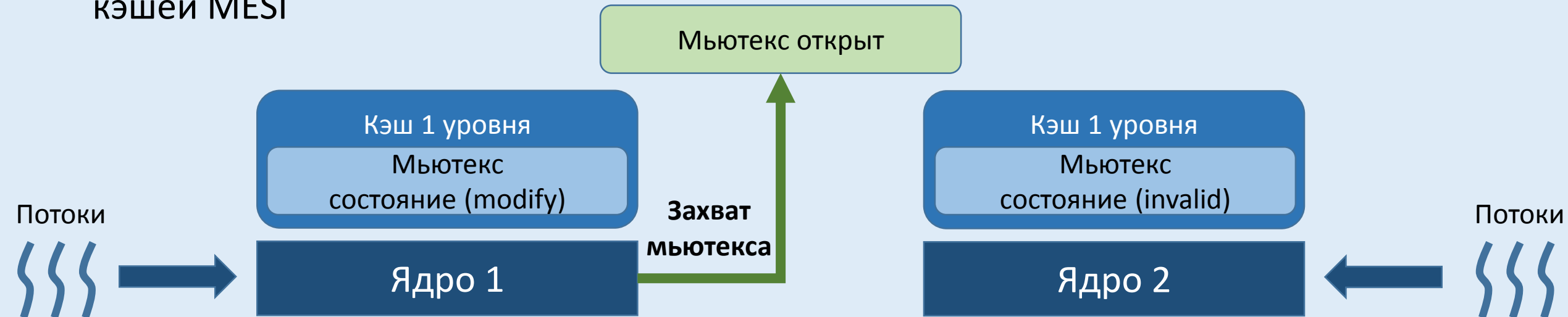
Конфликты

- Возникающие конфликты негативно сказываются на времени выполнения многопоточных программ в силу особенностей работы протокола когерентности кэшей MESI



Конфликты

- Возникающие конфликты негативно сказываются на времени выполнения многопоточных программ в силу особенностей работы протокола когерентности кэшей MESI



Алгоритм

- Цель алгоритма, реализованного в данной работе, — уменьшить время операции захвата мьютекса, учитывая статистику возникающих конфликтных ситуаций
- Алгоритм состоит из двух основных этапов
 - 1) Профилирование
 - 2) Оптимизация

Схематичное
представление алгоритма



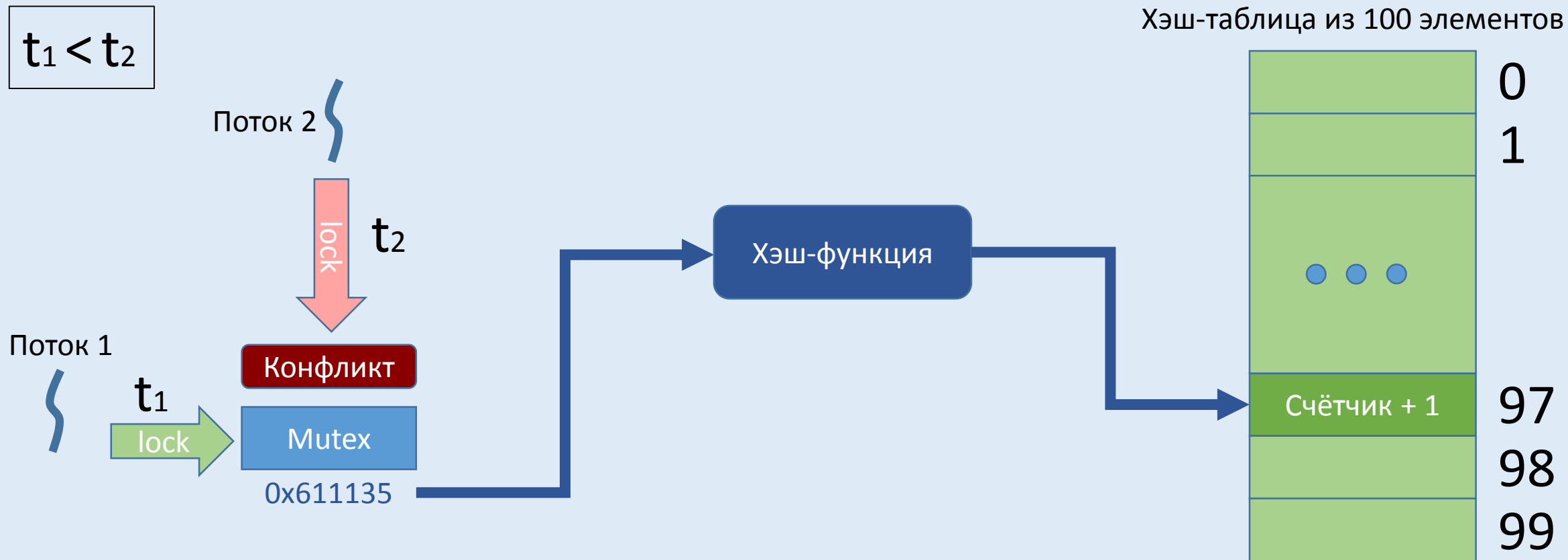
* Корректная работа алгоритма гарантируется только при совместном и поочерёдном выполнении обоих этапов (профилирование + оптимизация)

Профилирование

- На этапе профилирования производится подсчёт конфликтных ситуаций, возникающих при захвате мьютекса
- Вся информация записывается в хэш-таблицу, где ключом является адрес конкретного мьютекса, а значением – частота возникновения искомой ситуации для него

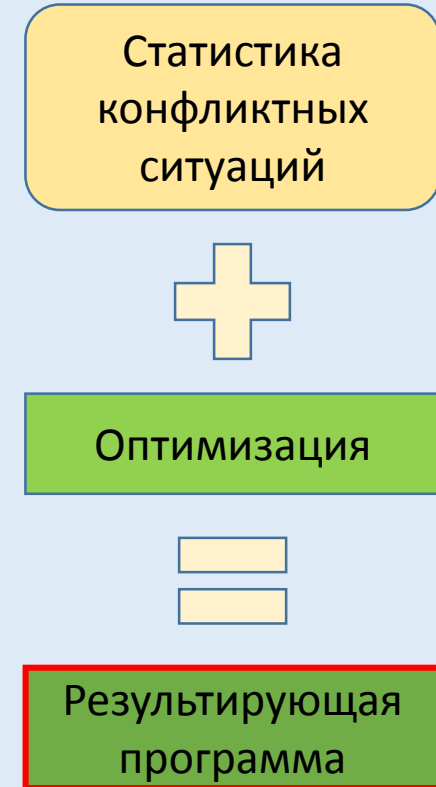


Профилирование



Оптимизация

- На данном этапе производится оптимизация «проблемных мьютексов», то есть таких, значение в хэш-таблице для которых больше среднего
- Иначе говоря, для мьютексов, за которые часто одновременно конкурируют множество потоков, регулируется показатель, отвечающий за количество попыток захвата, перед тем как отправить тот или иной поток «спать»



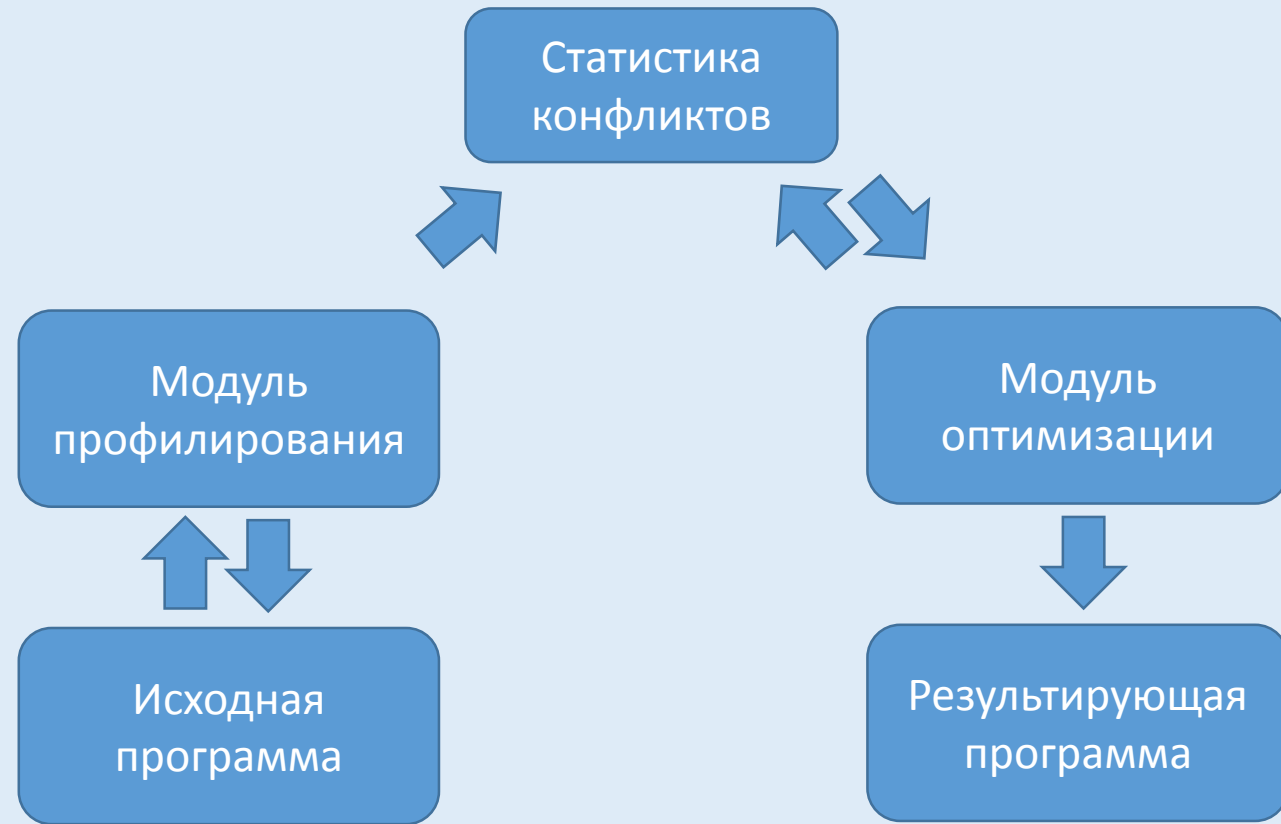
* В glibc 2.23 этот показатель является статическим и не регулируется до начала работы многопоточной программы

«mutex-optimizer»

- Функциональная структура программного пакета «mutex-optimizer»

1) Модуль профилирования

2) Модуль оптимизации



Результаты исследований

- Эффективность разработанного пакета исследовалось на синтетических тестах (microbenchmark)
- Тесты запускались на ноутбуке ASUS K53S под управлением ОС Linux (Fedora 21)

Показатели:

- Процессор: Intel Core i3 2350M (2.3 ГГц, 35 Вт)
- Количество ядер: 2
- Чипсет: Intel HM65 Express
- Оперативная память: 4 Гб SO-DIMM DDR3 1333 МГц

Результаты исследований

Входные данные

Количество мьютексов:

1

Количество потоков:

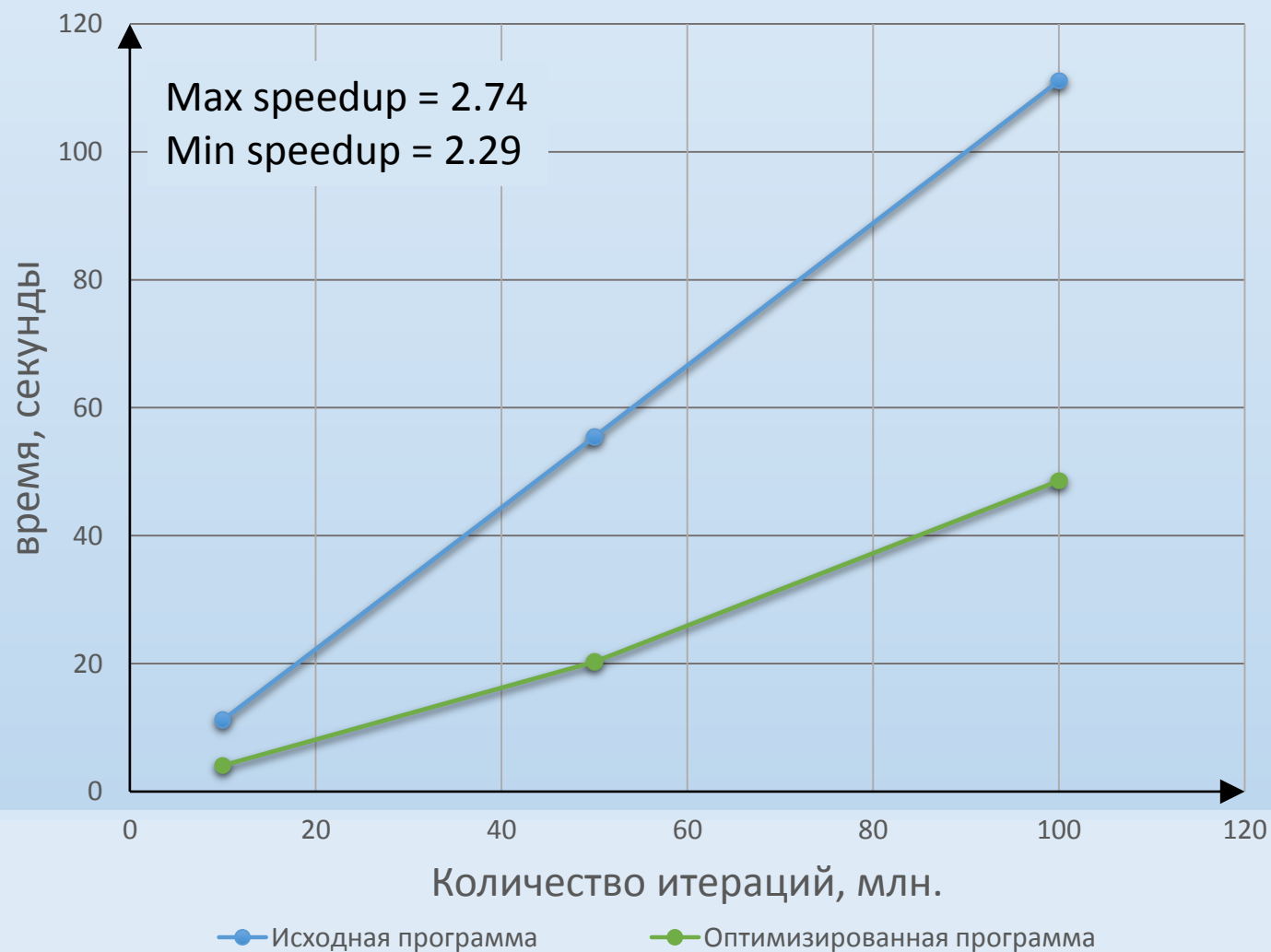
10

Количество итераций с входом
в критическую секцию:

1) 10 000 000

2) 50 000 000

3) 100 000 000



Результаты исследований

Входные данные

Количество мьютексов:

1

Количество потоков:

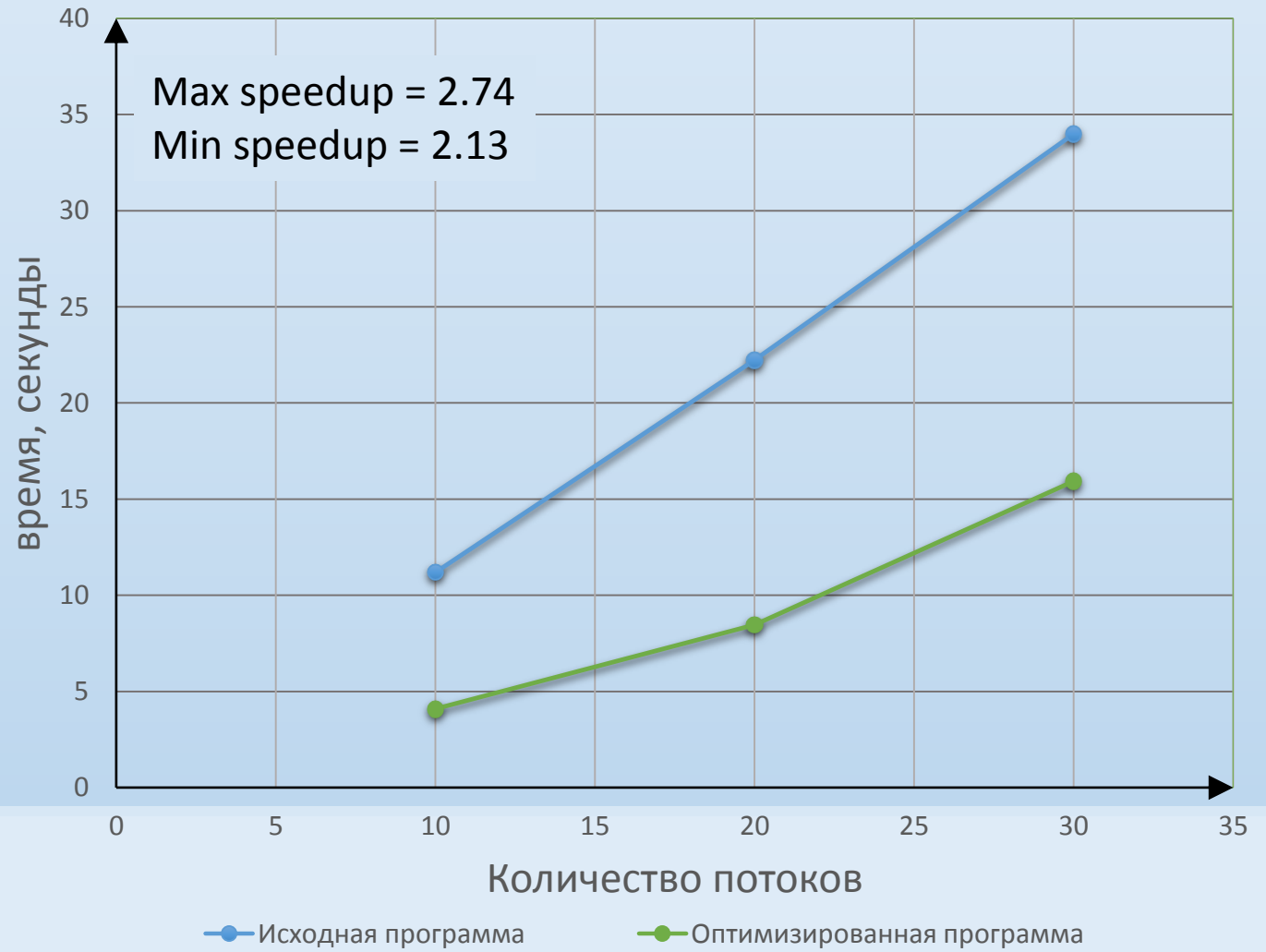
1) 10

2) 20

3) 30

Количество итераций с входом
в критическую секцию:

10 000 000



Работу выполнил:
Гайдай А. В.
Научный руководитель:
Кулагин И. И.

Спасибо за внимание!

Докладчик: Гайдай А. В.



Эл. почта: diligent20494@gmail.com
Телефон: +7 (983)-309-84-33