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

Адаптивный алгоритм операции захвата мьютекса

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

СибГУТИ

Мьютексы

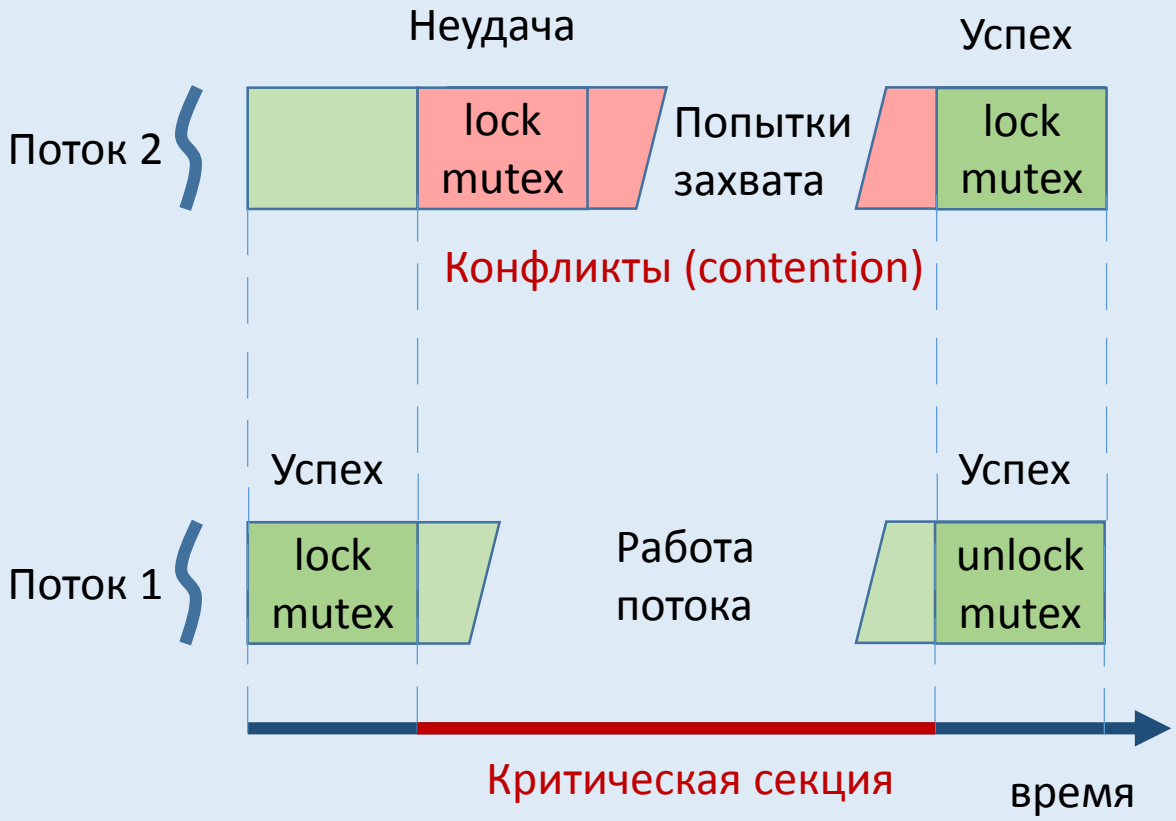
- Мьютекс – примитив синхронизации, позволяющий создавать в коде программы критические секции, выполнение которых возможно только одним потоком в каждый момент времени
- Мьютексы могут находиться в одном из двух состояний – открытом или закрытом

```
// 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);
```

СибГУТИ

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

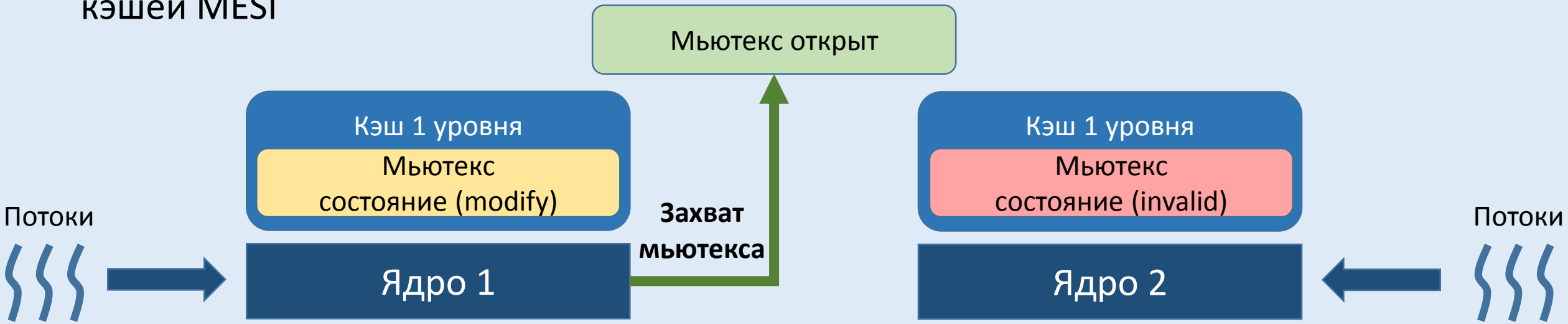
СибГУТИ

Конфликты

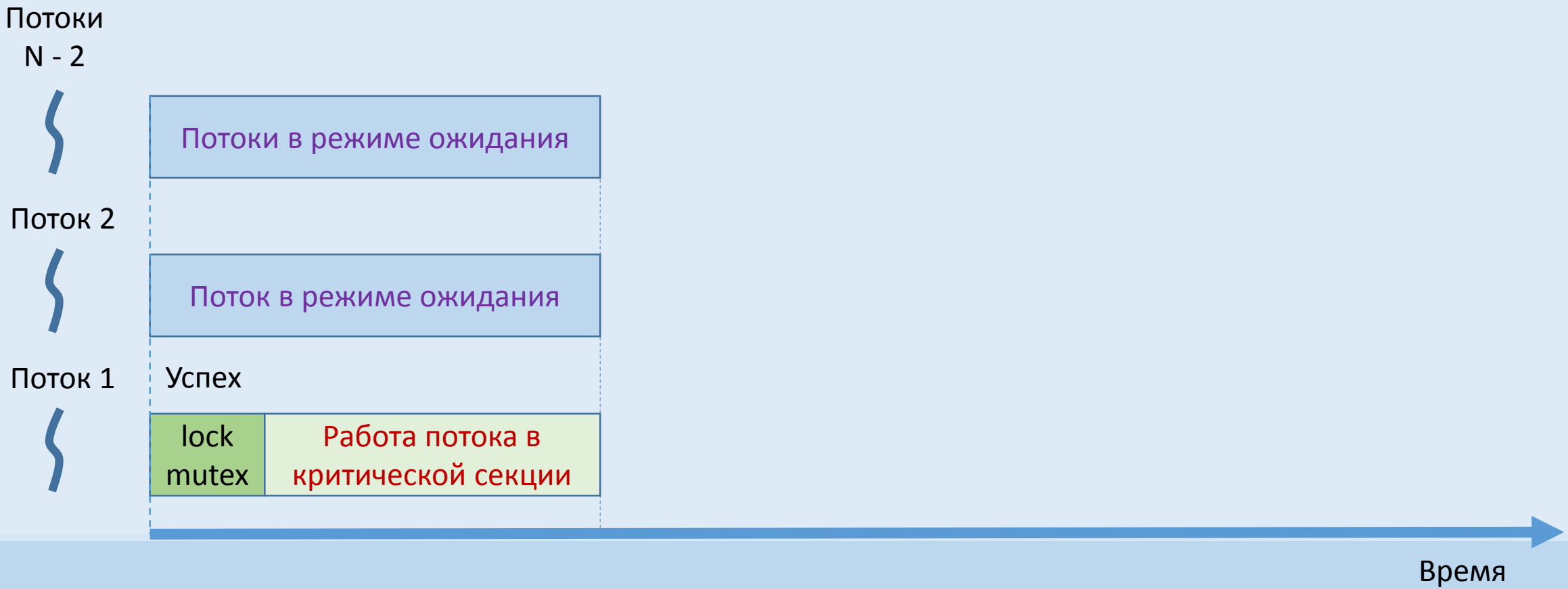
- Возникающие конфликты негативно сказываются на времени выполнения многопоточных программ в силу особенностей работы протокола когерентности кэшей MESI

Конфликты

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

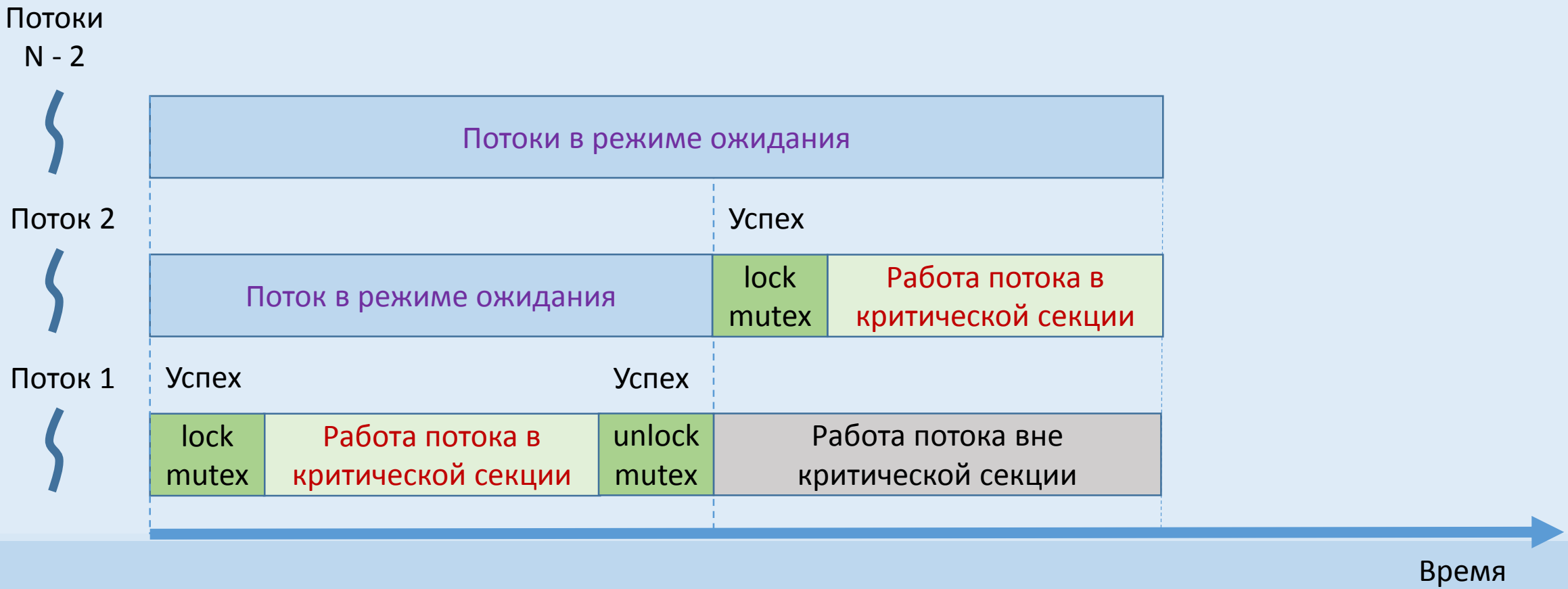


Идеальный случай



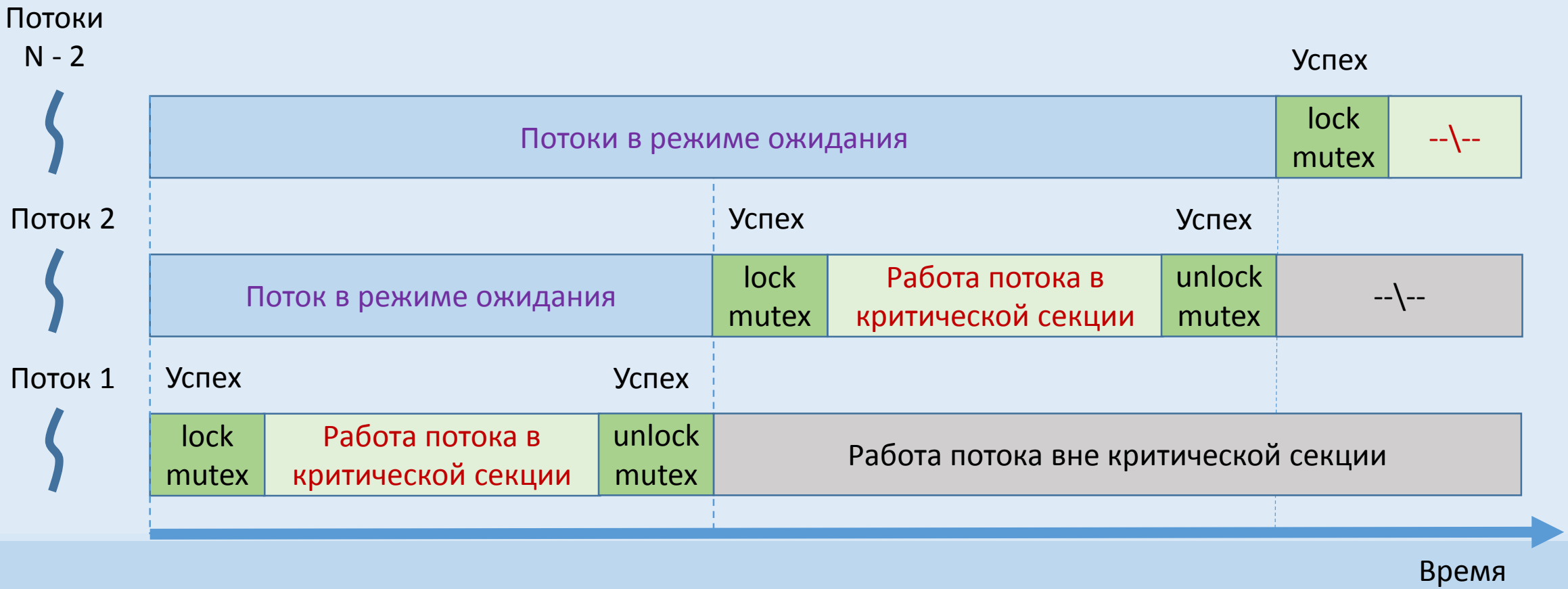
СибГУТИ

Идеальный случай



СибГУТИ

Идеальный случай



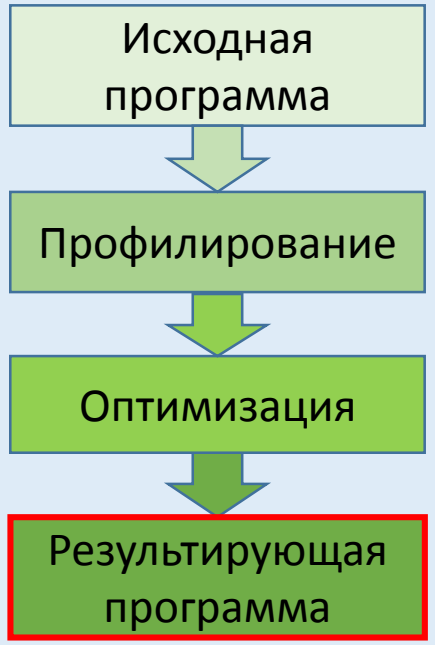
СибГУТИ

Алгоритм

- Цель алгоритма, реализованного в данной работе, — уменьшить время операции захвата мьютекса, учитывая статистику возникающих конфликтных ситуаций

- Алгоритм состоит из двух этапов
  1) Профилирование
  2) Оптимизация

Схематичное представление алгоритма



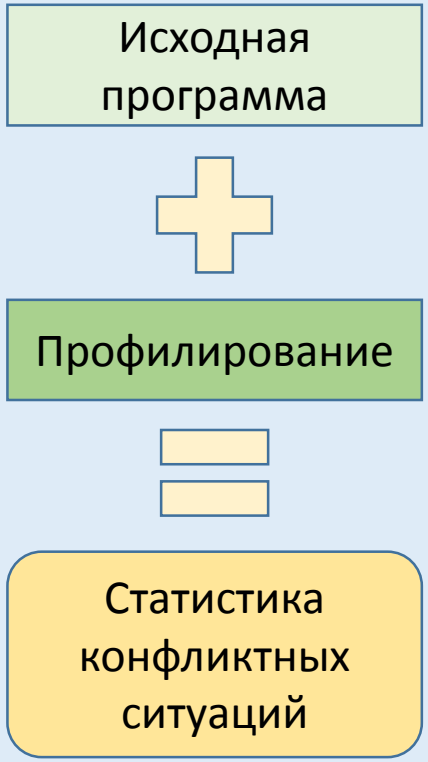
* Корректная работа алгоритма гаранируется только при совместном и поочерёдном выполнении обоих этапов (профилирование + оптимизация)

СибГУТИ

Профилирование

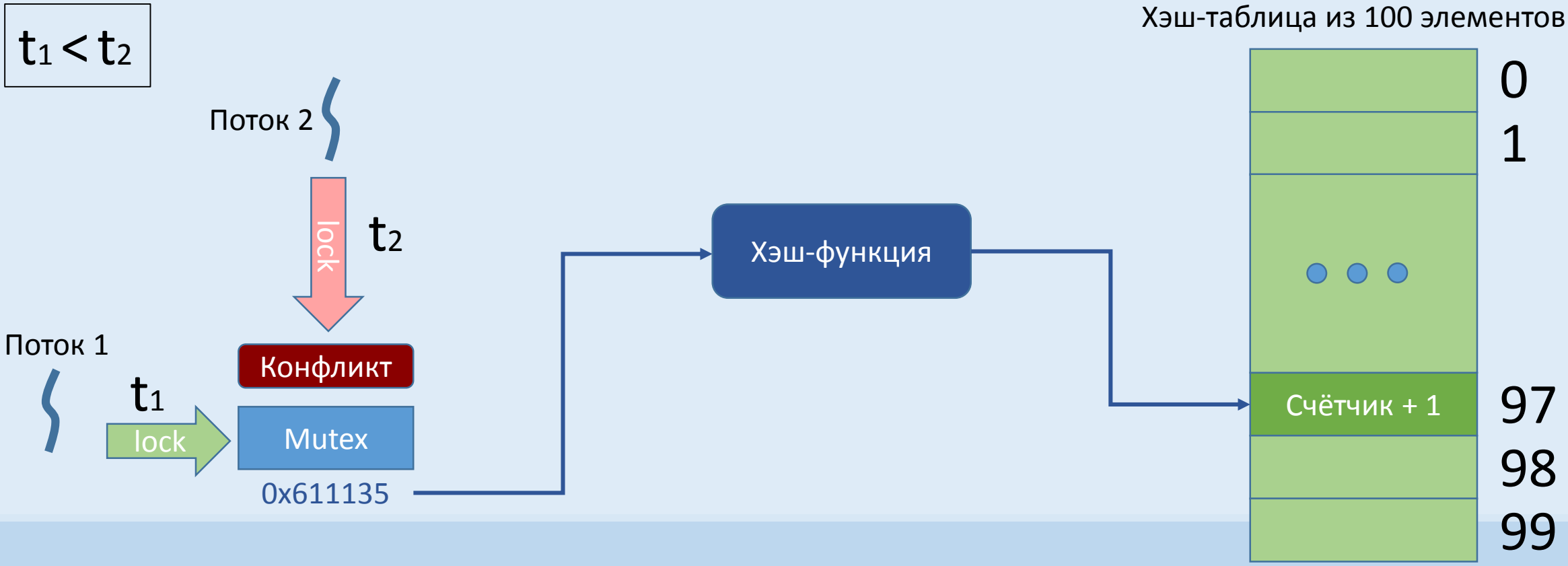
- На этапе профилирования производится подсчёт конфликтных ситуаций, возникающих при захвате мьютекса

- Вся информация записывается в хэш-таблицу, где ключом является адрес конкретного мьютекса, а значением – частота возникновения искомой ситуации для него



СибГУТИ

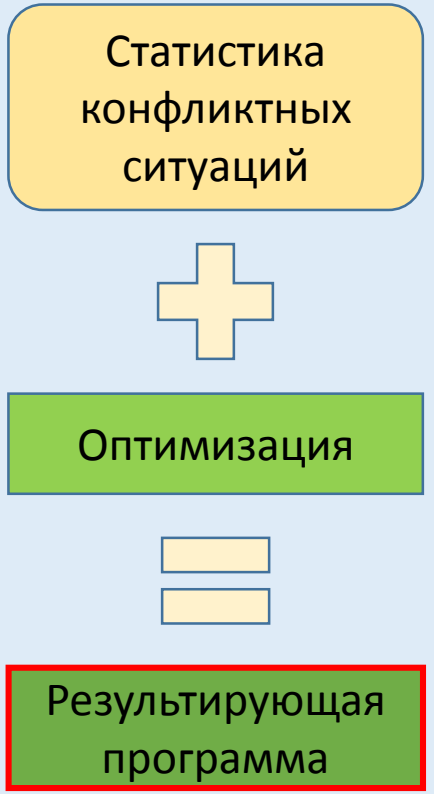
Профилирование



Оптимизация

- На данном этапе производится оптимизация «проблемных мьютексов», то есть таких, значение в хэш-таблице для которых больше среднего

- Регулируется показатель, отвечающий за количество попыток захвата, перед тем как отправить тот или иной поток в режим ожидания



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

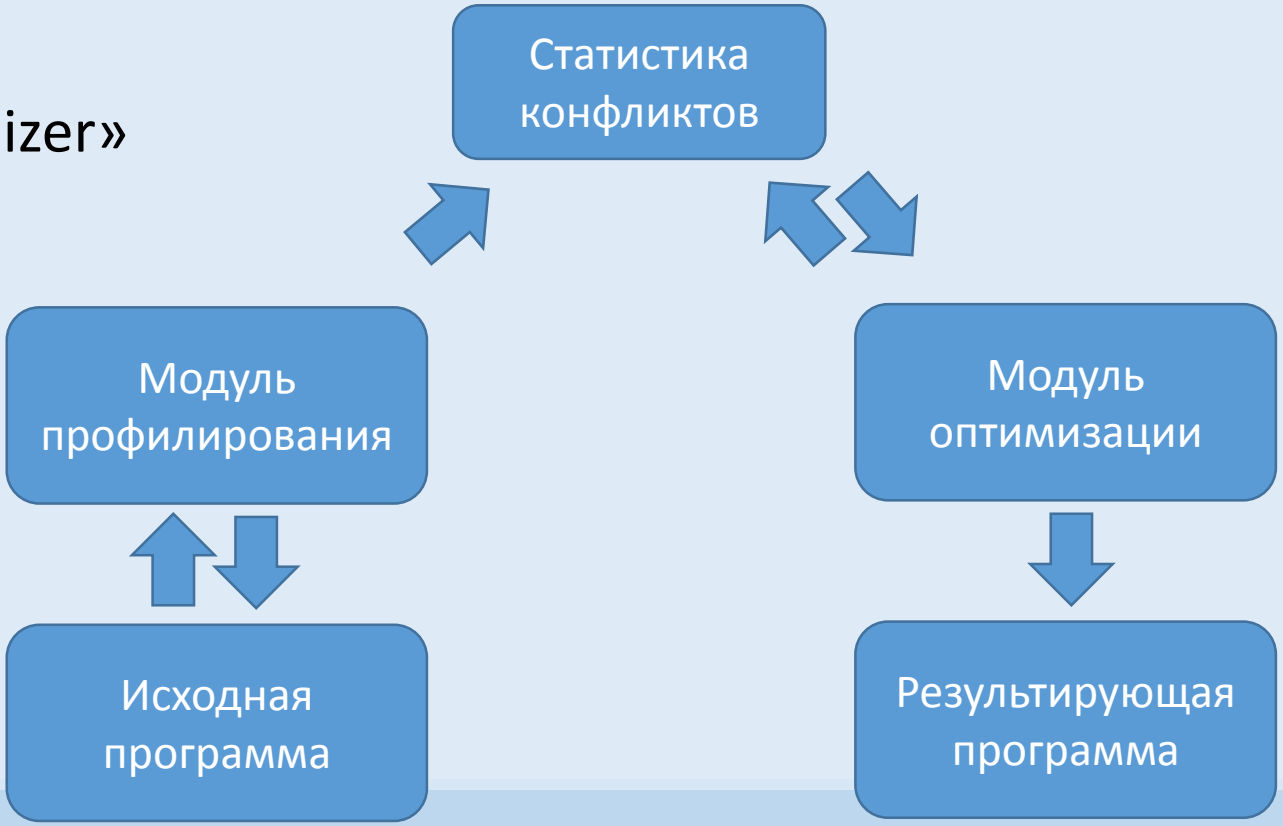
СибГУТИ

«mutex-optimizer»

- Функциональная структура программного пакета «mutex-optimizer»

  1) Модуль профилирования

  2) Модуль оптимизации



СибГУТИ

Результаты исследований

- Эффективность разработанного пакета исследовалось на синтетических тестах (microbenchmark)
- Тесты запускались на ноутбуке ASUS K53S под управлением ОС Linux (Fedora 22)
- GCC version 5.3.1

- Процессор: Intel® Core™ i5 2450M (2.5 ГГц, 35 Вт)
- Количество ядер: 2
- Оперативная память: 4 Гб SO-DIMM DDR3 1333 МГц

СибГУТИ

Результаты исследований

Входные данные

Количество мьютексов:

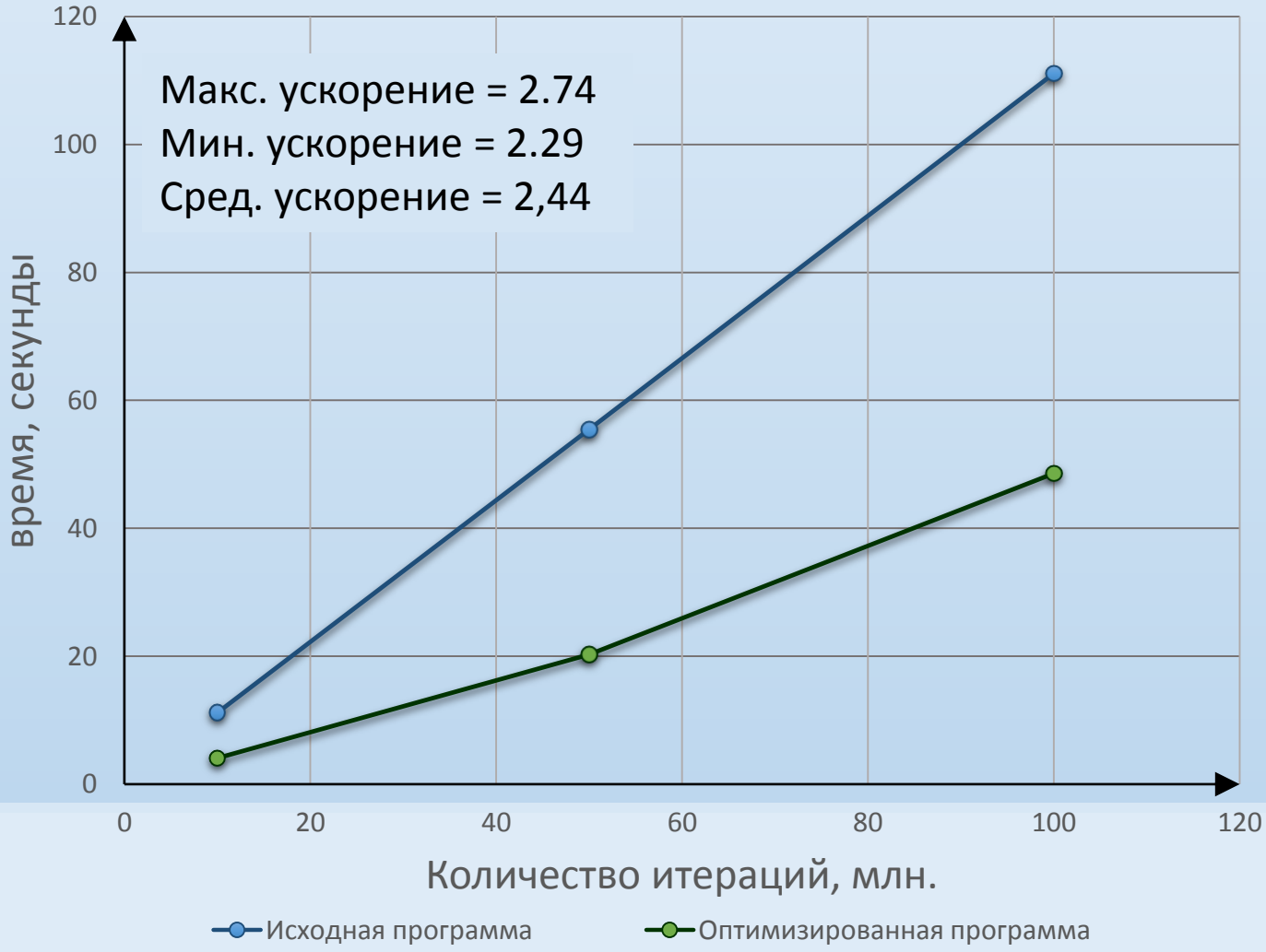
1

Количество потоков:

10

Количество итераций с входом в критическую секцию:

1) **10 000 000**
2) **50 000 000**
3) **100 000 000**



СибГУТИ

Результаты исследований

Входные данные

Количество мьютексов:

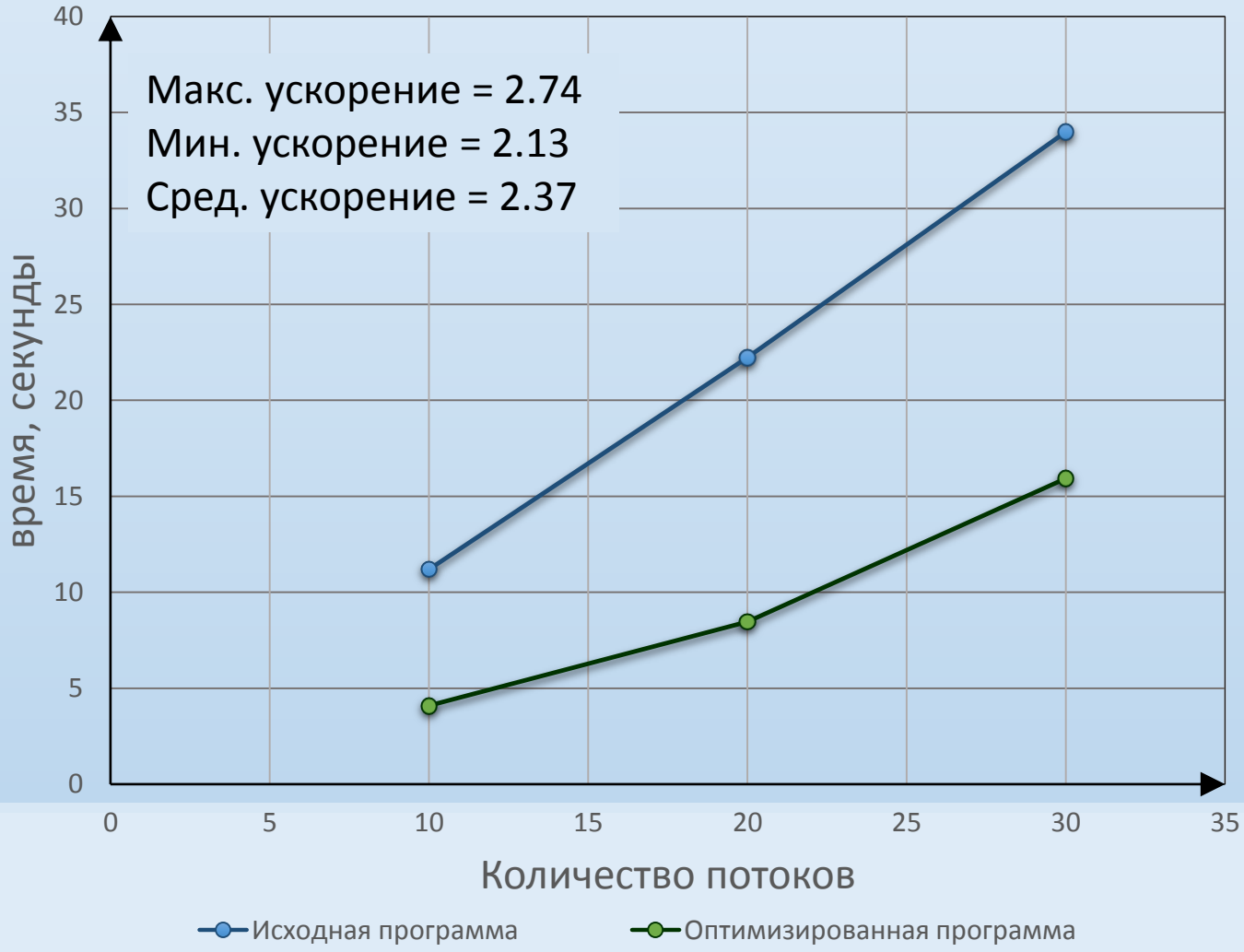
1

Количество потоков:

1) **10**
2) **20**
3) **30**

Количество итераций с входом в критическую секцию:

10 000 000



СибГУТИ

Результаты исследований

- Запуск тестов на кластере Jet
- ОС на вычислительном узле Linux (Fedora 20)
- GCC version 4.8.3

Вычислительный узел:

- Процессор: Intel® Xeon® CPU E5420 (2.50 ГГц)
- Количество ядер: 8
- Оперативная память: 8 GB (4 x 2GB PC-5300)

СибГУТИ

Результаты исследований

Входные данные

Количество мьютексов:

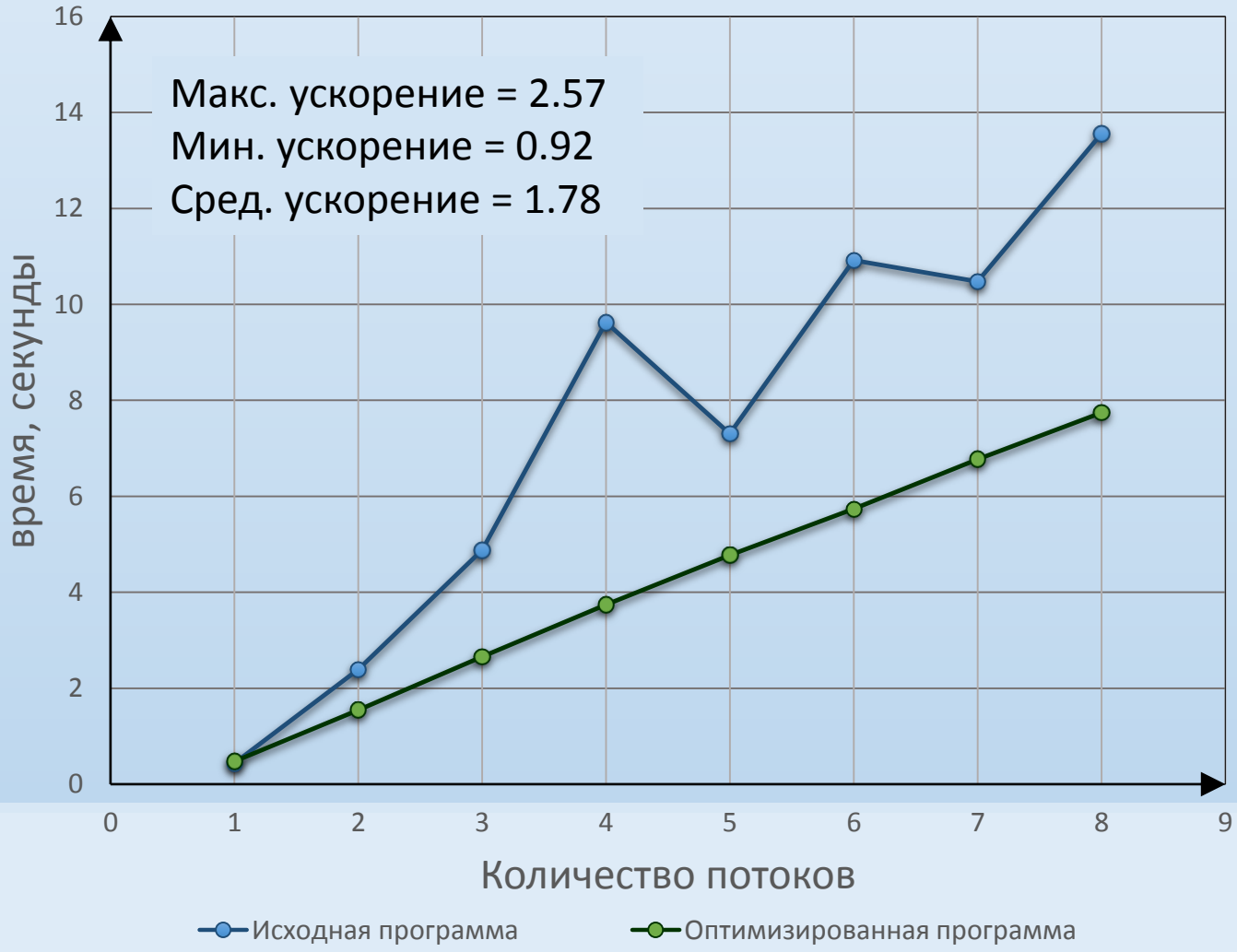
1

Количество потоков:

{ 1, 2, 3, 4, 5, 6, 7, 8 }

Количество итераций с входом в критическую секцию:

10 000 000



СибГУТИ

Результаты исследований

Входные данные

Количество мьютексов:

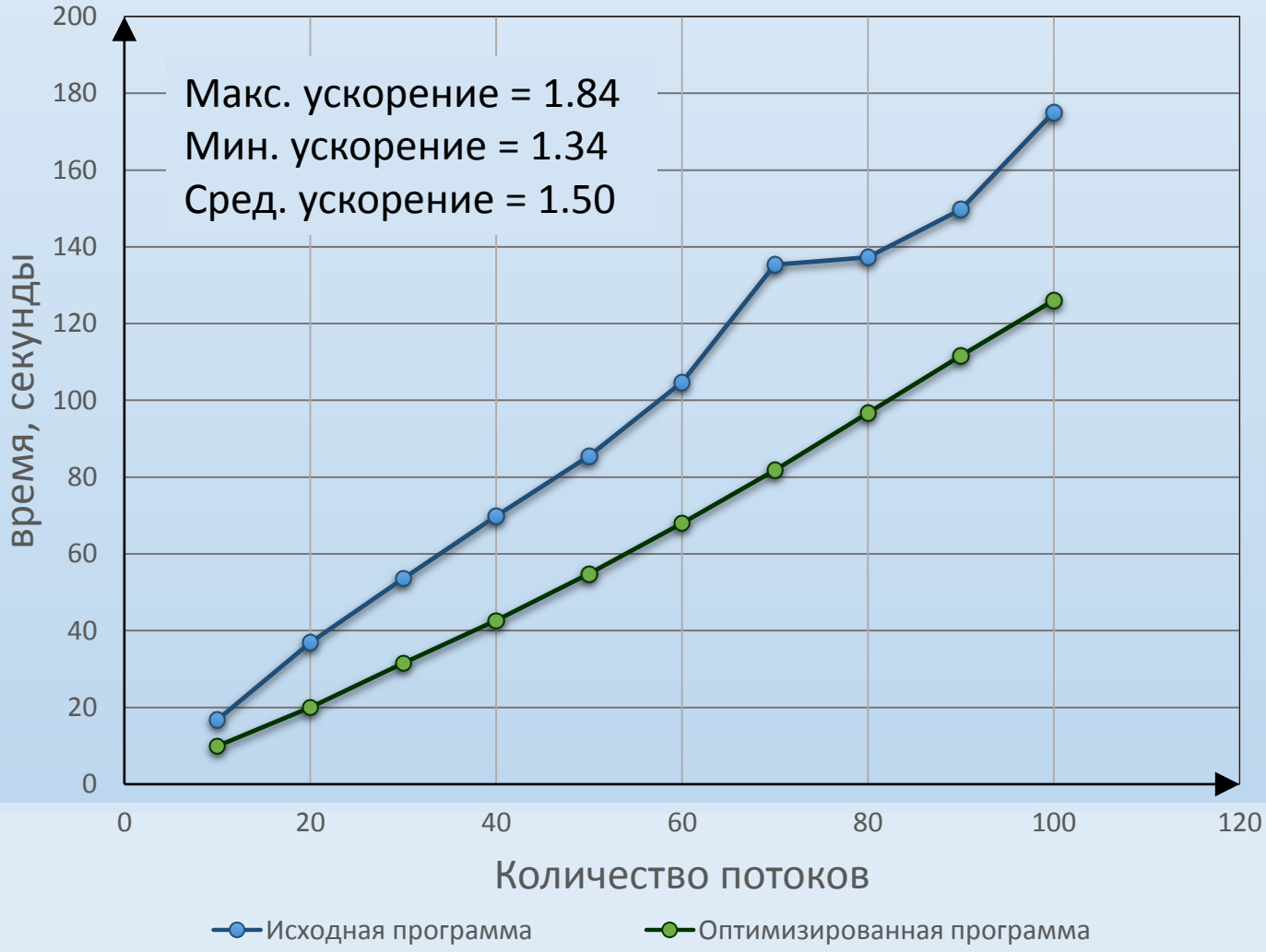
1

Количество потоков:

{ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 }

Количество итераций с входом в критическую секцию:

10 000 000



СибГУТИ

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