

---

# Параллельные структуры данных

Дамаскинский Константин

7 декабря 2021 г.

# Что требуется?

- Контейнеры
  - 1 Список (стек, очередь)
  - 2 Множество

# Модели доступа

- Single producer, multiple consumers
- Multiple producers, multiple consumers
- Single consumer, multiple producers

# Параллельное выполнение операций

- Чтение
  - 1 Чтение разрешено
  - 2 Изменение запрещено
- Изменение
  - 1 Чтение запрещено
  - 2 Изменение запрещено

От метода синхронизации доступа зависит, **на какую составляющую контейнера** накладываются указанные ограничения

---

Множество на связном списке

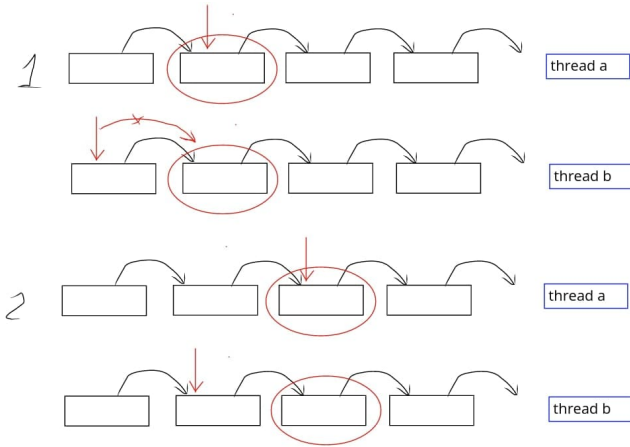
# Coarse-grained synchronization

- Блокирование доступа ко всему контейнеру при чтении (поиске элемента)
- Блокирование доступа ко всему контейнеру при записи
- Хорошо работает при небольшом количестве параллельных читателей
- Наименее эффективный метод синхронизации

# Fine-grained synchronization

- Блокирование доступа к каждому элементу списка по отдельности **по достижении**
- Позволяет итерироваться нескольким потокам одновременно
- Необходимо сначала получать доступ к очередному интересующему элементу, а затем освобождать доступ к текущему

# Fine-grained synchronization





# Optimistic synchronization

- **Проблема fine-grained sync.** Поток, который читает второй элемент, блокирует поток, который читает пятый элемент
- **Решение.** Контейнер не блокируется при поиске нужного элемента при чтении
- После нахождения требуемого элемента проверяется производится блокирование элемента. Затем проверяется, изменился ли элемент между моментом обнаружения и блокировки. Если не изменился, чтение успешно завершается

# Lazy synchronization

- Трудоёмкие операции выполняются в ленивом режиме. Конкретно: при удалении взводится флаг “удалён”, а физическое извлечение из памяти производится позже (после снятия блокировки)

---

# Lock-free очереди

# Lock-free и wait-free

- **Lock-free структура данных** гарантирует, что **некоторый** поток закончит выполнение операции над объектом за конечное число шагов *вне зависимости от результата работы других потоков* (даже если эти другие потоки завершились крахом)
- **Wait-free структура данных** гарантирует, что **все** потоки закончат выполнение операции над объектом за конечное число шагов

# Lock-free. Как это работает?

- Отказ от примитивов синхронизации
- Использование атомарных операций
  - 1 CAS – compare and swap  
`bool CAS(int *pAddr, int nExpected, int nNew)`
  - 2 TAS – test and set  
`int TAS(int *pAddr)`
  - 3 FAA – fetch and add  
`int FAA(int *pAddr, int nIncr)`

# ABA-проблема

- Поток 1: прочитал A
- Поток 2: заменил A на B
- Поток 2: заменил B на A

Возникает, когда первый поток держит указатель на удаляемый объект A, а в это время второй поток удаляет A, вставляет B, а затем опять возвращает A

# ABA-проблема

```
NodeType * Pop() {  
    Node * next ;  
    do {  
/*Pop1*/      NodeType * t = Top;  
/*Pop2*/      if ( t == null )  
/*Pop3*/          return null;  
/*Pop4*/      next = t->Next;  
/*Pop5*/  } while ( !CAS(&Top,t,next) );  
/*Pop6*/  return t;  
}
```

## Thread X

Вызывает Pop() .  
Выполнена строка Pop4 ,  
значения переменных: t == A  
next == A->next

## Thread Y

NodeType \* pTop = Pop()  
pTop == вершине стека, то есть A  
Pop()  
Push( pTop )  
Теперь вершина стека – снова A  
Заметим, что A->next изменилось

Выполняется строка Pop5 .  
CAS успешен, но полю Top->next  
присваивается значение,  
несуществующее в стеке,  
так как поток Y вытолкнул  
из стека A и A->next ,  
а локальная переменная next  
хранит *старое* значение A->next

## Как бороться? Отложенное удаление

# Особенности реализации lock-free очереди

- Общий концепт: пытаться прилинковать новый элемент к очереди в вечном цикле, пока CAS не завершится успехом
- Управление двумя указателями с помощью одного CAS (dCAS существуют не во всех реализациях x86-64)
- Для решения ABA-проблемы использовать, например, Epoch-based reclamation или Hazard pointer (ссылка)



# Источники

- M. Herlihy, N. Shavit, The Art of Multiprocessor Programming, Morgan Kaufmann, 2008
- Э. Уильямс, Параллельное программирование на C++ в действии, ДМК Пресс, 2012
- Атомарные примитивы (кликабельно)
- Lock-free стек (кликабельно)
- Lock-free очередь (кликабельно)