Построение многопоточных десктопных приложений на примере

Дмитрий Иванов, JetBrains

Многозадачность

- Многозадачнось **vs** Многопоточность
- Кооперативная **vs** Вытесняющая
- Процессы(Processes), потоки (Threads), задачи (Tasks)

Примитивы синхронизации

- `lock()`, Monitor, [Synchronized]
- EventWaitHandle: Manual, Auto
- Mutex – межпроцессное взаимодействие
- Barrier, CountDownEvent, Semaphore
- Slim, SpinWait – для короткого времени ожидания
- WaitHandle, Kernel32Dll

ReadWriteLock

- Всегда используйте Slim (или велосипед =)
- Upgradable – только один тред
- Fairness, starvation
- **using** вместо **try... finally**
- TryEnterReadLock, TryEnterWriteLock
- Номер транзакции WriteLock

Неблокирующие алгоритмы

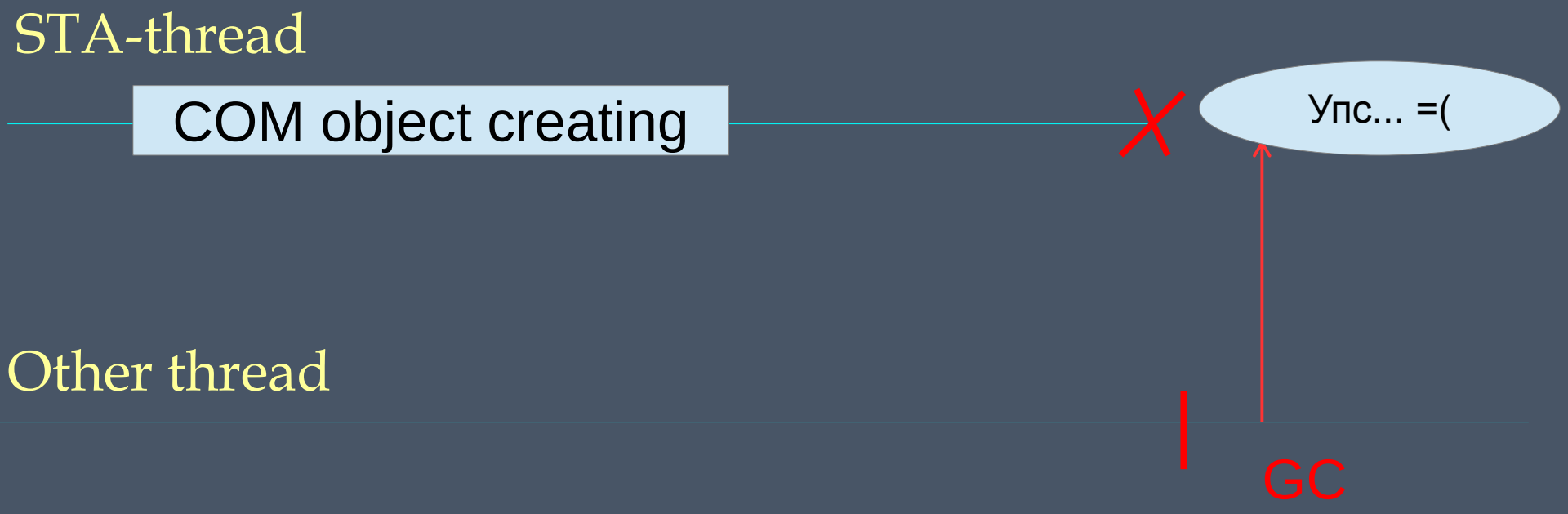
- **Interloked**: Read(Int64), Increment, Exchange, CompareExchange
- **Thread**: MemoryBarrier, VolatileRead
- Зачем:
  - Пишем свою структуру данных
  - Весело =)
- Общий вид алгоритма: `while (true)`
- **Тесты, тесты и еще раз тесты !!!**

SynchronizationContext

- [ThreadStatic]
- Send(), Post() (default → ThreadPool)
- Wait() (через WaitHelper)
- WPF → DispatcherSynchronizationContext

COM-объекты

- Appartments: STA, MTA, Neutral



Tasks

- Task(Action), Task<T>(Func<T>)
- short **vs** long
- Lifecycle (status): init → started → executing → completed (canceled, failed)
- Interruptable: CancellationToken
- Блокирующее API: Wait (One/All/Any)
- Task.Start()

Неблокирующее API

- StreamReader.ReadAsync() → Task<int>
- ContinueWith(), ContinueWhenAny()
- async /await – сахар для ContinueWith
- parent / child - неблокирующее ожидание

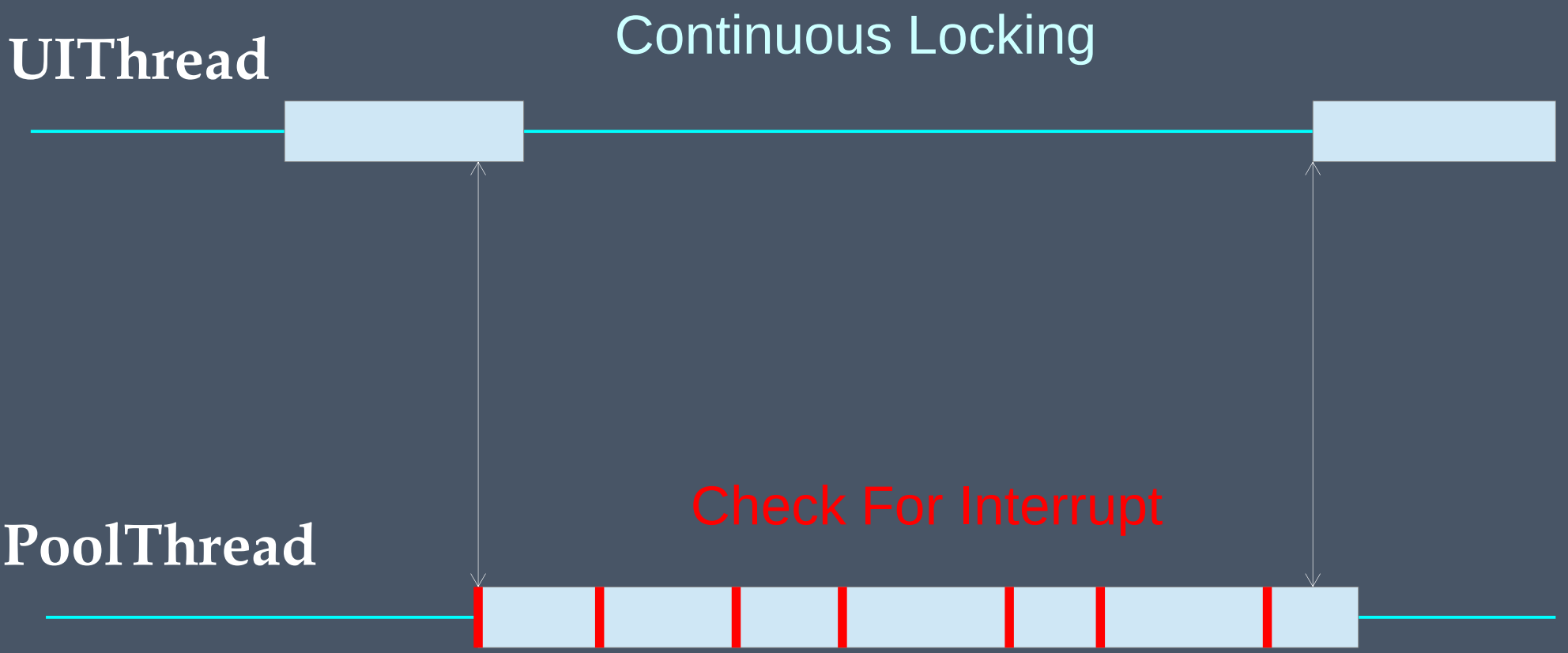
Модели многопоточного приложения

- Транзакционная
  - Immutable state
  - no locking
  - Commit/Rollback
- Пессимистичная
  - Глобальный RWLock
  - Прерываемые Read-ы
  - Проще, меньше memory usage

Модель ReSharper

- Main thread
  - Разрешен AcquireWrite
  - Разрешен Upgrade
  - Разрешен AcquireRead
- Pool thread
  - Разрешён TryAcquireRead
  - Везде CheckForInterrupt (<200ms)
  - Условно разрешён TryAcquireWrite

InterruptableReadActivity



GroupingEvent

prolongate

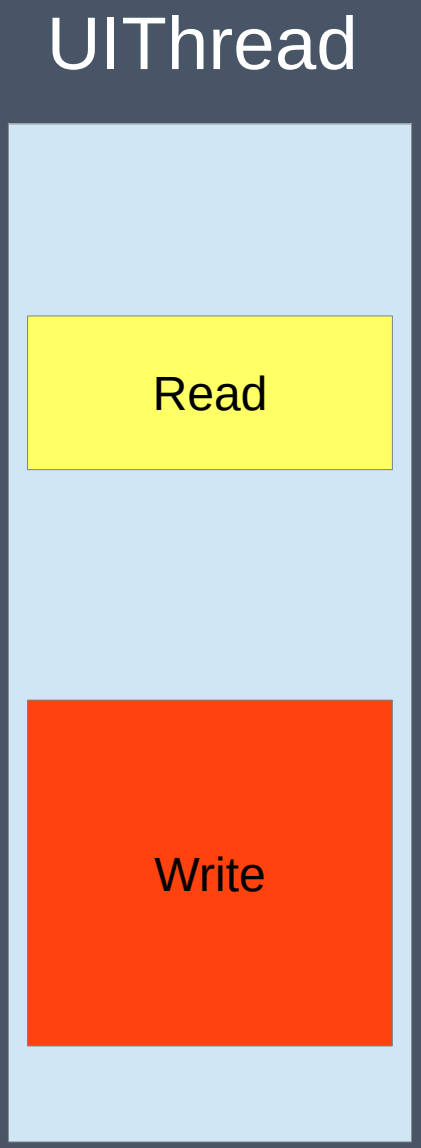
no prolongate

Другие конструкции

- SingleThreadExecutor
  - Очередь тасков
  - Возможно синхронное ожидаение
  - Stealing тасков
- TaskBarrier
  - Таски могут порождают таски
  - Топологическая сортировка зависимых тасков

ReentrancyGuard

- Reentrancy in UI Thread
- ExecuteOrQueue
- QueueWithReadLock, QueueWithWriteLock
- Dispatcher – вызов без гарда



Timer API

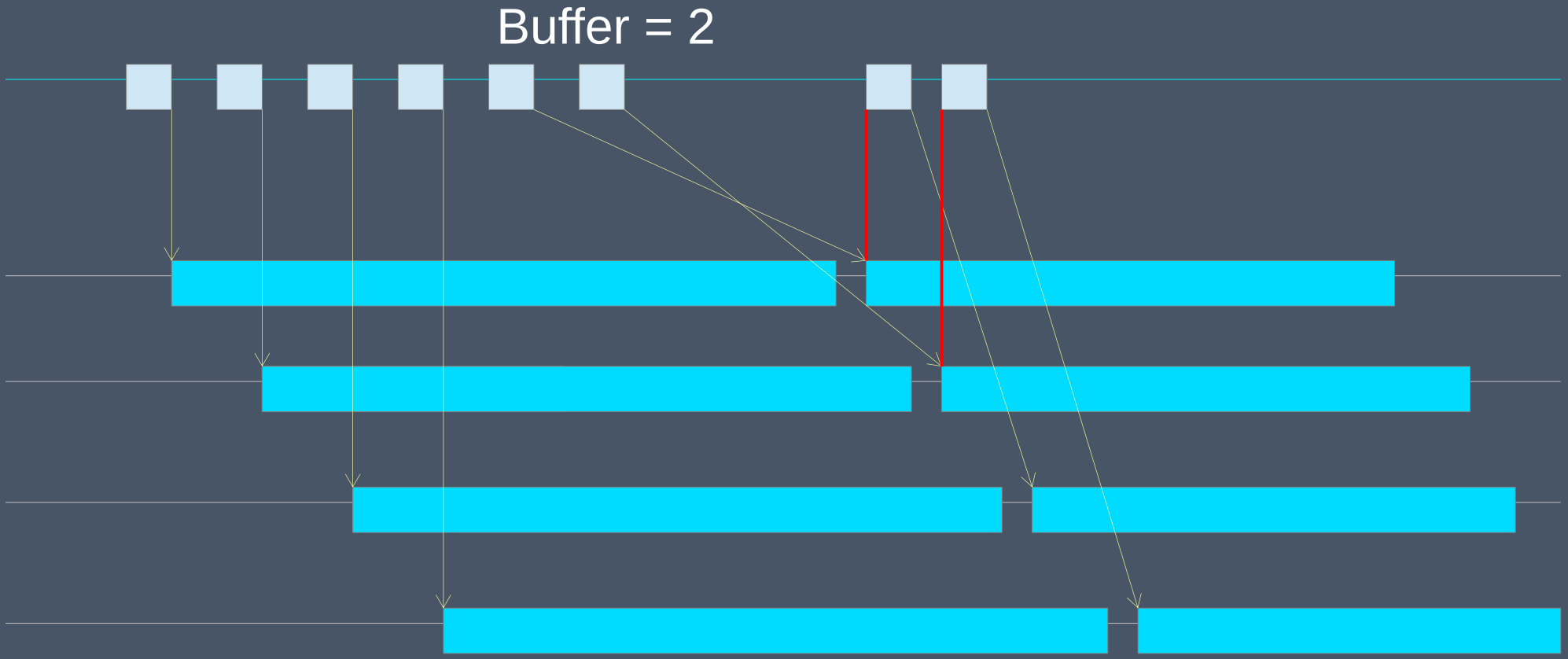
- QueueAt, QueueRecurring
- Время: UtcNow
- Самое быстрое: Environment.TickCount

IProperty<T>

- ISignal<T>
- FlowInto, CreateAnd(bool), Advise
- WhenAll
- FlowIntoGrouped - GroupingEvent
- FlowIntoUiThread – marshalling

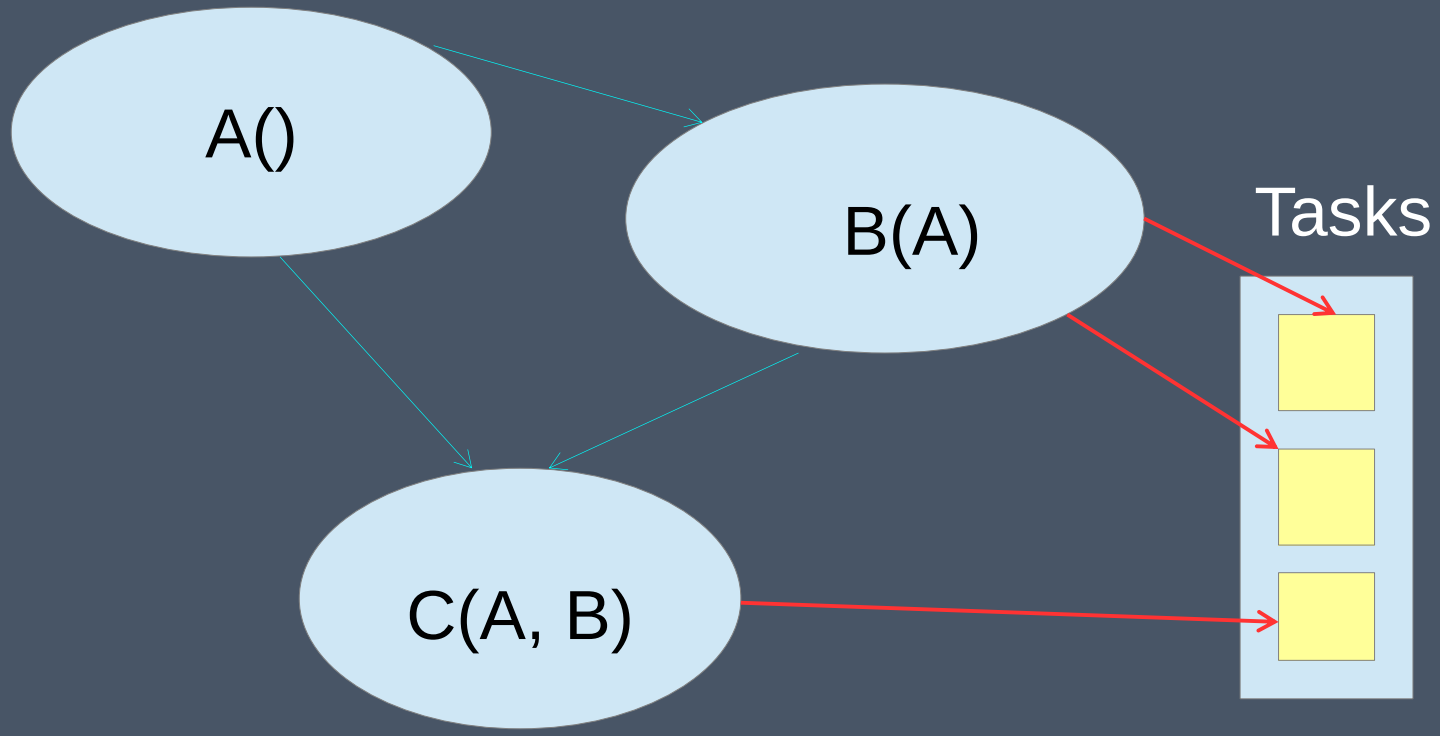
File IO

- PreprocessSingleThreadedAndParallelize



Component Container

- Инициализируется в UI-потоке
- Lifetime – инвертируем IDisposable



Средства анализа

- Видимые тормоза, дёрганность, лаги при тайпинге → подвисание UIThread (не качается очередь сообщений)
- Проверяем: между
  CheckForInterrupt < 200ms
- Проверяем: WriteLock.Acquire < 200ms
- **DotTrace Timeline**

Вопросы и ответы