3. Исследуйте производительность средств синхронизации при разном числе сообщений, разном объеме сообщений, разном числе потоков.

Параметры замеров времени: 10 читателей и 10 писателей (пишут по 5 строк)

**Без синхронизации** – долгое время: 20.3 сек. Повторы у читателей: 36, потери у писателей: 0.

При увеличении кол-ва сообщений, резко увеличивается повторяющиеся сообщения, значит, что потоки выполняют лишнюю работу. Это отражается на общем времени – слишком долго. Производительность плохая, а также гонка данных.

**lock** – самое долгое из синхронизированных: 18.2 сек, т.к. есть участки кода, где потоки не выгружаются, а делают в цикле пустую работу, ожидая изменения флага.

**AutoResetEvent, Semaphore** – примерно одинаковое: event: 0.101 сек., semaphore: 0. 164 сек. - меньше чем у lock. Потоки не делают бесполезную работу в ожидании.

**SemaphoreSlim** – хорошее время: 0.079 сек., хотя должно быть существенно лучше чем Semaphore, т.к. не вызываются функции ядра ОС – облегченный семафор. Надо проверить при других условиях.

**Interlocked.CompareExchange** – время: 1.9 сек. Время соптимизированно благодаря усыплению активно-ожидающих потоков на 50 мс. Если их не усыплять, то время доходило до 20 сек.

При увеличении числа сообщений – нет резкой зависимости. Резкое ухудшение есть при увеличении числа потоков у lock. Т.к. некоторые потоки остаются в активном ожидании.

4. Сделайте выводы об эффективности применения средств синхронизации.

В данной работе плохие результаты у lock, но не это не из-за проблем с самим lock, а из-за данного применения. Надо следить за тем, потоки выгружаются или нет. В данном случае выгружение дает заметную производительность.

**Вопросы и упражнения**

1. Почему проблема гонки данных проявляется не при каждом прогоне?

Бывает везет, и порядок выполения кода в потоках выполняется как будто синхронизованно, не прерывая друг друга.

2. Какие факторы увеличивают вероятность проявления проблемы гонки данных?

Б*о*льшее количество операций по обращению к общим данным (чтение, запись) в том участке, кода, который надо синхронизировать. Б*о*льшее количество потоков.

3. Возможно ли в данной задаче при отсутствии средств синхронизации возникновение исключения и аварийное завершение программы?

Был случай – когда для записи в общую коллекцию использовал List<string> - было исключение. Более не наблюдалось.

4. Можно ли в данной задаче использовать атомарные операторы для обеспечения согласованности доступа? Необходимы ли при этом дополнительные средства синхронизации?

Можно применить Interlocked.CompareExchange в качестве условия в блок if – тело блока if будет работать, как критическая секция. Проблема в том, что другие ожидающие потоки в цикле while постоянно будут проверять этот if.

5. Можно ли в данной задаче использовать потокобезопасные коллекции для обеспечения согласованного доступа?

Использовать их без средств синхронизации не достаточно. Я их использовал для записи общего результата со всех потоков – для сбора статистики об ошибках. Без потокобезопасных коллекций были баги – была неверная информация – гонка данных.

6. Какие средства синхронизации обеспечивают наилучшее быстродействие в данной задаче? Объясните с чем это связано.

Лучшие средства AutoResetEvent, Semaphore, SemaphoreSlim – обеспечивают выгруз блокированных потоков, которые ожидают доступа к критической секции. Выгруженные потоки не занимаются бесполезной работой по проверке флага во время ожидания, как в случае с lock.

На некоторых прогонах SemaphoreSlim показывает лучше результаты по времени, чем Semaphore – так и должно быть, т.к. он не вызывает функции ядра ОС.

[Комментарии (1)](https://dl.spbstu.ru/mod/assign/view.php?id=36736)

* [Изображение пользователя Туральчук КонстантинТуральчук Константин](https://dl.spbstu.ru/user/view.php?id=1167&course=272) - Чт, 12 мар 2020, 19:25

Почему нет атомарных операторов? С помощью оператора ComapareExchange можно обеспечить взаимно-исключительный доступ к произвольному фрагменту кода

Добавил п.4. Время выполнения можно улучшать опытнум путем с помощью подбора времени усыпления активно-ожидающих потоков.