1) *Объясните, почему захват и освобождение описанного спинлока промахиваются по кэшу лишь константное число раз.*

Захват и освобождение описанного спинлока промахиваются по кэшу лишь константное число раз, потому, что число промахов при захвате и освобождении данного спинлока не зависит от числа потоков, которые хотят захватить спинлок, и не зависит от параметров процессора, докажем, что это число действительно константное.

- В функции ***void Acquire***() может произойти промах по ключу, когда мы читаем is_owner _ в цикле *while (!is_owner_.load()) {},* менять значение is_owner_ может только один поток, и следовательно промахнемся тоже один раз, при этом придется брать обновление значение в основной памяти.
- В функции ***void Release()*** аналогично может проиойти промах по ключу в цикле *while (this->next_.load() == nullptr) {}* , ожидая, когда поток, которые добавился в очередь после нас, установит ссылку next_ , пока это произойдет мы читаем nullptr из кэша, когда поток поменяет, придется один раз обратится в память.

2) *Почему он не подвержен проблемам cache ping-pong или thundering herd?*

Не подвержен проблемам Cache ping-pong потому, что в циклах while мы только читаем значения, запись происходит лишь несколько раз. Thundering herd так же не возможен потому, что мы будим только один поток, устанавливая ему флаг is_owner_.

3) *Объясните, почему TAS, TATAS и Ticket спинлоки не гарантируют константного числа промахов по кэшу?*

- TAS
  Промахи по кэшу будут происходить в цикле while, когда будем пытаться захатить спинлок. Вызывая exchange мы производим бессмысленное сбрасывание кэш-линий других потоков, что не гарантирует константного числа.
- TATAS
  Данный спинлок сбрасывает у всех потоков (пытающихся взять лок) кэшлинию при выходе из критической секции, следовательно, число промахов будет пропорционально числу потоков
- Ticket
  Опять же не гарантирует константного числа промахов, так как при увеличении числа потоков, число промахов будет расти, так как при выходе из критической секции поток будет инвалидировать кэшлинию потов, которые ждут своей очереди.