*Невроз – это неспособность переносить неопределенность.*

*© Зигмунд Фрейд*

В своей статье "The Problem with Threads" от 2006 года Эдвард А. Ли критикует потоки как основное средство параллелизма. Он считает доминирующую сейчас модель вычислений для параллельных систем несовершенной, неудобной и небезопасной из-за недетерминированности этой системы. Ли иллюстрирует это тем, как обычная последовательная модель выполнения, которая по сути является детерминированной, с несколькими потоками становится не просто недетерминированной, но экспоненциально сложной, потому что невозможно быть уверенным, что каждое возможное чередование машинных инструкций по-прежнему будет выполнять правильные действия – выполнения одной и той же программы становятся неэквивалентными – и из этого автор делает вывод о несостоятельности потоков как модели параллельных вычислений. Обнаружив проблему, Эдвард А. Ли предлагает «альтернативы» потокам: рандеву, PN Director, SR Director и DE Director, рассказывает про MapReduce, акторы и передачу сообщений. Конечно, там тоже есть недетерминизм, но он, по словам автора, намного более правильный. Одним из способов победить ветряную мельницу многопоточности Ли видит координационные языки, которые в прошлом слишком много раз терпели неудачу. Но он уверен: победив множество трудностей, такие языки придут к успеху и «скорее всего, заменят традиционные языки».

С моей точки зрения, идеи, которые несет Эдвард А. Ли, несколько противоречивы. В своем предостережении из недалекого прошлого, когда многопроцессорные системы только входили в моду, он обвиняет неправильных программистов, пишущих неправильный код на неправильных языках программирования, работающих на неправильных потоках в том, что они используют неправильные технологии вместо того, чтобы весь мир потоков разрушить до основанья, а затем… Эти размышления Эдвард Ли сдабривает попыткой «строгого доказательства» потоков как бесполезной математической модели вычислений (как писал Гёте, «Математики похожи на французов: чтобы вы ни сказали, они всё переведут на свой собственный язык. Получится нечто противоположное»), и критикует наш хаотичный мир в целом. Ли пугливо предсказывает, как многопоточное программирование сделает компьютеры бесполезными (ведь программы на них будут часто падать), как координационные языки программирования заменят традиционные, и какой грустной будет жизнь продавцов компьютеров в этом антиутопичном будущем победивших потоков. Впрочем, глупо винить человека из прошлого в том, что реальность разошлась с его фантастическими предсказаниями.

Можно долго размышлять о проблемах параллелизма. И, хотя я во многом не согласен с выводами автора, статья несет некоторые полезные идеи. Проблемы с многопоточностью были известны давно (даже до 2006 года). Многопоточное программирование, вероятно, является самым сложным решением проблемы параллелизма. По сути, это довольно низкоуровневая абстракция того, что на самом деле делает машина. Параллельное программирование сложно из-за наблюдаемого недетерминизма, но при использовании правильного подхода к данной проблеме и правильной библиотеки, поддерживающей этот подход, можно избежать многих проблем. Сейчас, как и в 2006 году, существует ряд подходов, таких как модель акторов или транзакционная память, или работа с неизменяемыми структурами данных, которые намного проще. Вот почему потоки, как кажется мне, стоит рассматривать как базовый строительные блоки, на которых могут быть построены более сложные абстракции (корутины, каналы, передача сообщений, структурированный параллелизм и т. д.). Возможно, об этом же хотел сказать и автор, но его фанатичный «джихад» и сквозящая между строк нелюбовь к устоявшимся способам программирования заставляют в этом усомниться и задуматься: не ради ли громкой и провокационной статьи Эдвард Ли предпочел здравые мысли фанатичным грёзам о «принципиально новой модели вычислений».

СИМВОЛОВ МАЛО, НЕОБРЕЗАННАЯ ВЕРСИЯ: <https://pastebin.com/4YfJfmiW>

*============ Uncutted Version =============*

*Невроз – это неспособность переносить неопределенность.*

*© Зигмунд Фрейд*

В своей статье "The Problem with Threads" от 2006 года Эдвард А. Ли критикует потоки как основное средство параллелизма. Он считает доминирующую сейчас модель вычислений для параллельных систем несовершенной, неудобной и небезопасной из-за недетерминированности этой системы. Ли иллюстрирует, как обычная последовательная модель выполнения, которая по сути является детерминированным, с несколькими потоками становится не просто недетерминированной, но экспоненциально сложной, потому что невозможно быть уверенным, что каждое возможное чередование машинных инструкций по-прежнему будет выполнять правильные действия – выполнения одной и той же программы становятся неэквивалентными – и из этого автор делает вывод о несостоятельности потоков как модели параллельных вычислений. Обнаружив проблему, Эдвард А. Ли предлагает «альтернативы» потокам: рандеву, PN Director, SR Director и DE Director, рассказывает про MapReduce, акторы и передачу сообщений. Конечно, там тоже есть недетерминизм, но он, по словам автора, намного более правильный. Одним из способов победить ветряную мельницу многопоточности Ли видит координационные языки, которые в прошлом слишком много раз терпели неудачу. Но он уверен: победив множество трудностей, такие языки придут к успеху и «скорее всего, заменят традиционные языки».

С моей точки зрения, идеи, которые несет Эдвард А. Ли, несколько противоречивы. В своем предостережении из недалекого прошлого, когда многопроцессорные системы только входили в моду, он обвиняет неправильных программистов, пишущих неправильный код на неправильных языках программирования, работающих на неправильных потоках неправильных операционных систем, выполняющихся на неправильных компьютерах в том, что они используют неправильных технологии вместо того, чтобы весь мир потоков разрушить до основанья, а затем… Эти размышления Эдвард Ли сдабривает попыткой «строгого доказательства» потоков как бесполезной математической модели вычислений (как писал Гёте, «Математики похожи на французов: чтобы вы ни сказали, они всё переведут на свой собственный язык. Получится нечто противоположное»), и критикует наш недетерминированный и хаотичный мир в целом, пытаясь отобразить множество бабочек на множество циклонов в Индонезии – иными словами, загоняет свои аргументы в прокрустово ложе демагогии, пытаясь выставить их в удобном для себя ключе. Автор пытается показать более красивые и эффективные, чем потоки, модели параллелизма, использующие под капотом, как ни странно, потоки – автор несколько отрекается от своих радикальных идей. Ли пугливо предсказывает, как многопоточное программирование сделает компьютеры бесполезными (ведь программы на них будут часто падать), как координационные языки программирования заменят традиционные, и какой грустной будет жизнь продавцов компьютеров в этом антиутопичном будущем победивших потоков. Впрочем, глупо винить человека из прошлого в том, что реальность разошлась с его фантастическими предсказаниями.

Можно долго размышлять о проблемах параллелизма. И, хотя я во многом не согласен с выводами автора, она несет некоторые полезные идеи. Проблемы с многопоточностью были известны давно (даже до 2006 года). Многопоточное программирование, вероятно, является самым сложным решением проблемы параллелизма. По сути, это довольно низкоуровневая абстракция того, что на самом деле делает машина. Но автор предпочитает бороться с ветряными мельницами недетерминированности, забывая, что правильное разделение задач упрощает многопоточность. Это то, о чем часто забывают люди, когда создают 20 потоков, пытающихся обработать один и тот же буфер. Если у вас есть блокировка в логике вашего приложения, вы делаете что-то не так. Параллельное программирование сложно из-за наблюдаемого недетерминизма, но при использовании правильного подхода к данной проблеме и правильной библиотеки, поддерживающей этот подход, можно избежать многих проблем. Сейчас, как и в 2006 году, существует ряд подходов, таких как модель акторов или транзакционная память, или работа с неизменяемыми структурами данных, которые намного проще. Вот почему потоки, как кажется мне, стоит рассматривать как базовые строительные блоки, на которых могут быть построены более сложные абстракции (корутины, каналы, передача сообщений, структурированный параллелизм и т. д.). Возможно, об этом же хотел сказать и автор, но его фанатичный «джихад» и сквозящая между строк нелюбовь к устоявшимся способам программирования заставляют в этом усомниться и задуматься: не ради ли громкой и провокационной статьи Эдвард Ли предпочел здравые мысли фанатичным грёзам о Франкенштейне «принципиально новой модели вычислений».