В статье «The Problem with Threads» Эдвард А. Ли критикует одну из самых популярных моделей параллелизма – потоки. Их использование сравнимо со сборкой двигателя из хаотично движущихся молекул: в теории возможно, на практике – сложно и ненадежно. Потоки удобно внедряются в уже существующие языки программирования, но нарушают детерминизм: из-за произвольного порядка выполнения операций одинаковые входные данные могут дать разный результат. При этом даже после тщательной разработки и тестирования все еще могут вылезти ошибки. Так Ptolemy II, разработанный командой Ли с применением строгого контроля качества и множества тестов, поймал дедлок через четыре года эксплуатации.

Ли также рассматривает альтернативы потокам — CSP, Kahn Process Networks, синхронные языки и т.д. Они более предсказуемы, но редко используются, т.к. не вписываются в привычную последовательную парадигму программирования.

В заключение, вместо борьбы с недетерминизмом, автор предлагает строить параллелизм на основе уже детерминированных моделей и использовать языки координации, которые описывают взаимодействия между частями программы и работают поверх уже существующих языков.

На мой взгляд, идеи Ли звучат убедительно, т.к. он подкрепляет их примерами из практики (Ptolemy II, работа с потоками на Java, схемы и описания моделей). Еще статья помогает частично избежать типичных для многопоточности ошибок и знакомит нас с альтернативными способами организации параллелизма, что, безусловно, полезно.

Параллельные вычисления становятся важнее из-за распространения многоядерных процессоров, распределённых систем и задач машинного обучения. Но мы продолжаем стабильно сталкиваться с описанными Ли проблемами: гонки данных, дедлоки, сложности тестирования кода. Значит и статья сохраняет свою актуальность.

Более того, многие идеи Ли уже получили свое развитие. Например, язык Go строит параллелизм на основе CSP. Растёт популярность и других подходов: реактивное программирование, dataflow-графы (TensorFlow) и даже фреймворки вроде React используют принципы, близкие к тем, что описывал Ли — например, однонаправленный поток данных вместо хаотичного изменения состояния.

И все же, на мой взгляд, подход автора слишком радикален: идея полностью отказаться от потоков и строить параллелизм с нуля звучит красиво, но очень нереалистично. Потоки сейчас используются почти повсеместно. Сможем ли мы переписать весь этот код на новые модели? Это нереально.

При этом, если посмотреть на области, где важна точность и предсказуемость — такие, как промышленные или авиационные системы — там уже давно используются другие подходы, например синхронные языки вроде Lustre или Esterel.

В массовой же разработке мне видится другой путь — гибридный. Мы можем использовать потоки там, где они удобны, и одновременно внедрять более предсказуемые модели там, где нужна надёжность.

Поэтому я склоняюсь к тому, что будущее не в отказе от потоков, а в их модернизации. Индустрия выбирает эволюцию, а не революцию — и я согласна: порой лучше немного улучшить старое, чем полностью менять на новое. Возможно, именно гибридные решения станут тем самым балансом между надёжностью и практической применимостью.