В статье Эдварда Ли "The Problem with Threads" анализируются недостатки существующего метода организации параллельных вычислений, основанного на потоках.

В первой части доклада, начиная с предисловия, Эдвард Ли подробно рассматривает проблему надежности в многопоточном программировании. Он показывает, насколько этот подход нестабилен и как негативно он влияет как на разработчиков и их приложения, так и на конечных пользователей. Подчеркивается, что зачастую важнее обеспечить стабильность и предсказуемость работы программ, чем добиться максимальной производительности и гибкости; предлагается иметь возможность добавлять недетерминированность при необходимости, а не устранять её по мере необходимости. В ходе объяснения автор отмечает, что поточные компоненты крайне недетерминированы по природе, и на примере реализации паттерна "наблюдатель" на Java демонстрирует, насколько сложными или вовсе неочевидными могут быть решения даже простых задач в многопоточном контексте. Автор, как мне показалось, с юмором и некоторой иронией говорит о разработчиках многопоточного кода, называя их безумцами, и дает четкое определение безумия — "это повторение одних и тех же действий снова и снова в надежде изменить результат".

Во второй части доклада Эдвард рассматривает методы и подходы, которые позволяют хотя бы частично бороться с многочисленными ошибками, такими как исключения и deadlocks. Он ссылается на проект Ptolemy Project и отмечает — несмотря на высокое качество разработки — полностью протестировать систему практически невозможно или очень сложно из-за её сложности. В ходе обсуждения он затрагивает темы транзакций, как с программной, так и с аппаратной стороны; рассказывает о координационных языках и расширениях существующих языков программирования, таких как C, C++ и Java. Также он рассматривает концепции promises и futures, которые помогают управлять асинхронностью. В завершение Эдвард вновь обращается к теории вычислений и предлагает формальные модели для описания параллельных вычислений.

Итак, проанализировав этот очень увлекательный доклад, хочу поделиться своими мыслями. Трудно не согласиться с автором в том, что параллелизм действительно представляет собой серьёзную проблему. Добавление любой абстракции в программировании часто ведёт к уменьшению полного понимания происходящих процессов. Когда вы вводите в код различные классы, интерфейсы, методы с разным уровнем вложенности и другие языковые конструкции, вы повышаете уровень абстракции, отходя от конкретных функций, которые реально выполняет программа или её части. И это касается даже последовательного, однопоточного программирования. Однако с появлением параллелизма вероятность ошибок растёт стремительно, поскольку становится почти невозможно проследить логику выполнения и взаимодействие столь же абстрактных функций одновременно в нескольких потоках.

Также хочется добавить, что любая система должна стремиться к максимальной надежности, насколько это возможно; однако использование параллелизма снижает и её прозрачность, и общую надёжность, что негативно сказывается на конечном результате в той или иной степени. Конечно, стоит отметить, что полностью отказаться от параллелизма в современном мире практически невозможно, поскольку такие компании, как Intel и другие производители многоядерных процессоров, активно продвигают идеи многопоточного программирования и развития параллельных технологий.

Сейчас мы живем в то время, когда технологии развиваются с невероятной скоростью, но тем не менее вряд ли можно будет найти такое решение, которое удовлетворило бы все требования. Вместо этого нужно стремиться к последовательному улучшению текущих

подходов и поиску новых эффективных решений, которые помогут разработчикам создавать более надежные и производительные параллельные программы.