Concurrency != Parallelism

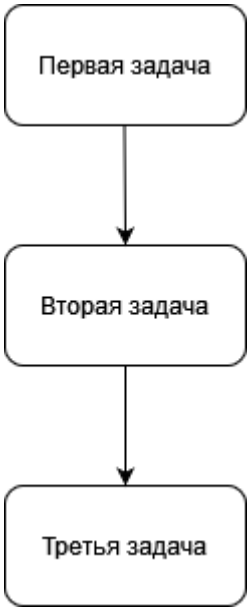
Серышев Денис

Оглавление

Введение

Зачастую разработчики путают понятия concurrency(конкурентности) и параллелизма, предполагая, что они являются одним и тем же, но это не так.

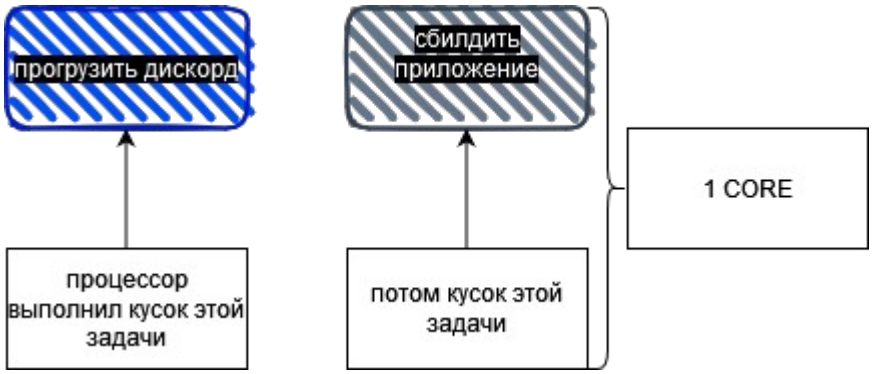
Начнем с последовательного подхода. Тут идет выполнение одной задачи в один момент времени. То есть, таск за таском. Сначала мы с друзьями говорим в дискорде, дорабатываем пет-проект, пьем чай, смотрим видео на ютубе. Недостатком данного подхода является то, что мы эффективно неактивны(заблокированы) в момент выполнения задачи. Мы не можем во время выполнения какой-то задачи, например, общения с друзьями в дискорде так же писать код для нашего пет-проекта. Есть-ли способ как-то быстрее выполнять данные задачи? Мы же можем пить чай и смотреть видео на ютубе, разве нет?



На схеме выше изображен последовательный подход к написанию кода. Сначала у нас выполнится первая задача, потом переход ко второй, потом переход к третьей.

Concurrency

Предположим у нас старенький ноутбук с одноядерным процессором и мы хотим выполнять более, чем одну задачу одновременно(говорить с друзьями в дискорде и писать код), но как это сделать? Ведь одно ядро может выполнять только одну операцию в единицу времени? Следовательно, мы не в состоянии выполнять несколько задач одновременно. Спешу вас обрадовать. Мы сможем общаться с нашими хорошими друзьями и писать наш чудесный код. Но как? Наш процессор будет переключаться между задачами. Он выполнил кусок какой-то задачи(в нашем случае прогрузил дискорд),переключился на другую задачу, например сбилдил нашу программу, потом опять переключился. После выполнения куска какой-то задачи он возвращается к первой задаче и тд. Эти две задачи запущены в двух разных потоках(threads). Переключение между потоками происходит настолько быстро, что создается впечатление, что эти потоки выполняются одновременно/параллельно, но это не так. Мы просто не замечаем разницы. То есть, ***конкурентно*** – это когда у нас ***N*** потоков конкурируют за один процессор.



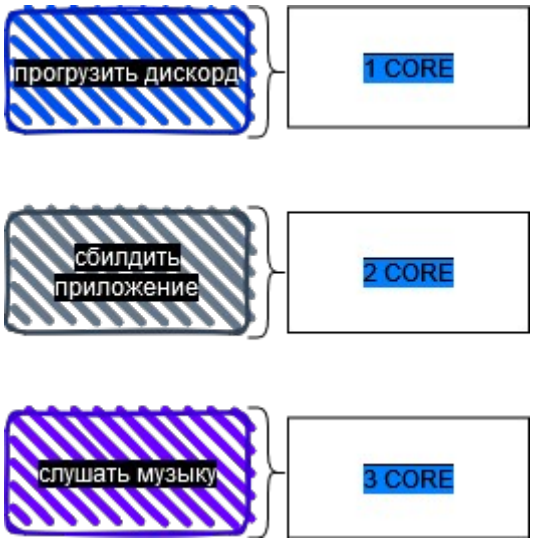
Как показано на схеме выше, данный подход позволяет нам разбивать задачи на блоки, которые могут выполняться независимо друг от друга и общаться между собой.

Когда у нас происходит переключение потоков, это называется *Context Switch.* При переключении с одного потока на другой, ОС меняет контекст текущей задачи на контекст следующей задачи следующего потока.

Конкурентность – это **взаимодействие** с несколькими задачами одновременно.

Параллелизм

В основном на серверах используется процессор от компании Intel, а именно Xeon, но почему? Их же не просто так называют “многоядерными монстрами”, но что имеется ввиду под “многоядерными”? Чем больше ядер, тем больше работы процессор способен осуществлять. На каждом ядре будет выполняться какая-то задача. В нашем же примере на первом ядре мы будем говорить с друзьями, на втором - дорабатывать наш пет-проект, на третьем ядре пить чай, на четвертом смотреть видео. И все это будет выполняться одновременно. То есть, многоядерный процессор способен обрабатывать несколько инструкций в единицу времени, по одной инструкции на каждое ядро.



(примеры отличаются для понятливости, потому что процессор обрабатывать питье чая не может)

Параллелизм – это **выполнение** нескольких задач одновременно.

Итог

Подведя итог можно сделать вывод, что конкурентность не равна параллелизму. Конкурентность – это *взаимодействие* с несколькими задачами одновременно, а параллелизм – это *выполнение* нескольких задач одновременно.