

През: 8 мин - вмн. / 5 мин - регуляторам

① Аномалии. 1,5 - 2 смр.

(Ани. есть все, но
без симметрии)

| Технк: 21 зевр. предпомощительно.

② Обзор предгеморроидальной области. 10-20 смр.

③ Венм и постоянная загад. Требование к
реализации 2 - 3 смр.

④ Решение и описание его.

Минимум описание нужна решения.

④* Тестирование и блокирование.

Ногтевидный, это регуляторам ног - это
использоваться (документы)

? Заключение - 1 смр.

Что сказать? Куда пойти?

Объем: 30 - 40 смр. min 25 смр.

СОДЕРЖАНИЕ

АННОТАЦИЯ	6
ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ	7
0.1 Котлин и C#	9
0.2 Язык Go	9
0.3 Проект "Loom"	9
ЦЕЛИ И ЗАДАЧИ	10
ОПИСАНИЕ РЕШЕНИЯ	11
ЗАКЛЮЧЕНИЕ	12
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	13

АННОТАЦИЯ

Сопрограммы (или корутины) с точки зрения пользователя языка программирования - это потоки, которые управляются средой исполнения языка или самим программистом. Преимущество корутин перед потоками операционной системы заключается в потреблении меньшего объема системных ресурсов – адресного пространства и памяти. Кроме того, переключение контекста корутины дешевле, чем потока, поскольку осуществляется средой исполнения языка.

В наше время усилился запрос разработчиков многопоточных веб-сервисов на поддержку корутин в языках программирования, поскольку они позволяют упростить и ускорить разработку приложения из-за своих преимуществ, перечисленных выше. Потому сопрограммы были реализованы в популярных языках: Go, Kotlin, C# и других. К сожалению, сейчас не существует нативного решения в языке Java. Целю работы является создание работающего прототипа корутин в языке Java. Работа проводится на базе виртуальной машины Excelsior Research Virtual Machine.

ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Сопрограмма (с английского coroutine) — программная компонента, особым образом организованная для обеспечения взаимодействия с другими компонентами по принципу кооперативной многозадачности. Выполнение модуля может быть приостановлено в определённой точке и предано другой сопрограмме. При этом будет сохранено полное состояние сопрограммы (включая стек, значения регистров и счётчик команд).

Концепция сопрограмм не нова: впервые они появились в языках программирования Симула, Модула-2 в 1960-е — 1970-е годы и использовались как еще одно языковое средство для реализации итераторов, генераторов, бесконечных списков и так далее. К сожалению, в тот момент концепция не получила широкого распространения и в более поздних языках программирования, таких как Си, С++, Java, она не была применена. Если нужна альтернатива сопрограммам, то рекомендовалось использовать потоки.

Начиная где-то с 2010-го года все стало стремительно меняться. Программы все это время росли и появилась проблема масштабируемости таких приложений. Одним из способов масштабирования программ это параллелизм: если хотим обработать кусок данных, который может быть достаточно крупным, то мы можем распределить его обработку на несколько потоков. Но есть другой, более сложный и распространенный вид масштабирования, который касается одновременной обработки относительно независимых задач, требуемых от приложения - конкуренция(с англ. concurrency). То, что они должны обслуживаться одновременно, - это не выбор реализации, а требование.

Рассмотрим на примере веб-сервера. Каждый из запросов, которые он обслуживает, в значительной степени независим от других. Для каждого из них выполняется синтаксический анализ пакета, делается запрос к базе данных и/или к другому серверу, формируется ответ, который отправляется в клиенту. Каждый запрос не взаимодействует с другими одновременными HTTP-запросами, но они конкурируют с ними за процессорное время и ресурсы

ввода-вывода. Каждое параллельное приложение имеет некоторые единицы параллелизма, естественные для его области, причем исполнение некоторой работы выполняется независимо от другой в то же время. Для веб-сервера это может быть HTTP-запрос; для базы данных это может быть транзакция.

Проблема заключается в том, что поток, программная единица параллелизма, не может соответствовать масштабу естественных единиц параллелизма приложения - пользовательского сеанса, HTTP-запроса или транзакция в базу данных. Сервер может обрабатывать до миллиона одновременных открытых сокетов, но операционная система не может эффективно обрабатывать более нескольких тысяч активных (не бездействующих) потоков. Забавно, что потоки были придуманы для виртуализации скучных вычислительных ресурсов с целью совместного использования, сами стали дефицитными ресурсами. Поскольку создание новых потоков дорогостоящая операция, их объединяют в пул для дальнейшего переиспользования.

К сожалению, пул предлагает слишком грубый механизм разделения потоков. Часто в пуле просто недостаточно потоков для представления всех независимых задач, выполняемых одномоментно. Заимствование потока ОС из пула на все время выполнения задачи удерживает поток, даже когда он ожидает какого-либо внешнего события, например, ответ от базы данных, сервера, или любого другого действия, которое может его заблокировать. Потоки ОС слишком важны, чтобы за них можно было держаться, когда задача просто ждет. Чтобы совместно использовать потоки более точно и эффективно, стоило бы возвращать поток в пул каждый раз, когда задача должна ждать некоторого результата. Это означает, что задача больше не привязана к одному потоку для всего своего выполнения. Это также означает, что мы должны не допускать блокировки потока, потому что такой поток станет недоступен для любой другой работы.

Решением проблемы является использование асинхронного API и неблокирующего ввода/вывода. Эти механизмы позволяют вернуть поток в пул, пока задача ждет результата операции. Программы, реализованные в таком стиле сложнее отлаживать, профилировать и искать в них ошибки. Но по другому

поступить нельзя, если нужно удовлетворить требование масштабируемости приложения и необходимо эффективно использовать текущие аппаратные ресурсы. Некоторые языки программирования борются со сложностями асинхронного кода.

0.1 Котлин и C#

В языках программирования JavaScript и C# реализовали концепцию "async/await".

0.2 Язык Go

0.3 Проект "Loom"

ЦЕЛИ И ЗАДАЧИ

ОПИСАНИЕ РЕШЕНИЯ

ЗАКЛЮЧЕНИЕ

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Каталог программных продуктов семейства Oracle Database [Электронный ресурс] // Oracle Россия URL: http://oracle.ocs.ru/files/catalog_Oracle_Database_12C.pdf (дата обращения: 13.04.2015)