слайд 1.

В современном мире язык Java используется для создания крупных многопоточных приложений для решения многих задач, в том числе и научных. Например, Java используется для создания систем управления Большим Адронным Коллайдером и для параллельной обработки результатов экспериментов, например, в библиотеке Colt Parallel. Язык Java выбран не спроста, поскольку это язык с управляемой средой исполнения, что значетельно упрощает разработку.

слайд 2

Традиционно, параллелизм реализуется операционной системой с помощью механизма потоков, которые абстрагируют иногда взаимодействующие между собой независимо работающие задачи. В современных реалиях количество потоков обычно превышает число ядер процессора, аппаратного механизма параллелизма. Операционным системам требуется представлять потоки как универсальное средство параллелизма для все языков программирования, потому потоки дорогие: создание, переключение потоков несет в себе крупные накладные расходы. Это становится особенно заметно c ростом числа потоков в программе.

Но существует возможная альтернатива потокам - сопрограммы.

слайд 3

<фактически пересказываю, что написано на слайде>

слайд 4

Сопрограммы уже реализованы в языках программирования С++ стандарта 20, С# и в относительно молодом языке Go. К сожалению, на сегодняшний день сопрограммы не реализованы в языке Java, но работы в этом направлении ведутся проектом "Loom".

слайд 5

<пересказ>

слайд 6

Целью же моей работы является изучение применимости сопрограмм как альтернативу потокам в программах Java. Для этого были поставлены следующие задачи: <слайд>

слайд 7

Первым шагом работы стала разработка тестов производительности различных реализаций сопрограмм и потоков. Анализ предметной области показал, что не существует подходящего набора тестов, который бы измерял 3 интересующих параметра:

1) Накладные расходы на создание потока/сопрограммы. Поскольку они пропорциональны времени создания, то измеряется время.

2) Скорость переключения и потребление физической памяти.

Тесты доступны по ссылке на слайде.

слайд 8

Следующим шагом работы стала реализация переключения контекста сопрограмм. Переключение - это процесс передачи управления от одной сопрограммы другой в точках явного планирования, помеченных на слайде функцией yield(). Так же переключение происходит при завершении исполнения сопрограммы.

слайд 9

к переключению контекста есть несколько подходов. Для реализации в HuaweiJDK использовался механизм как в языке Go, поскольку он показался более эффективным.

слайд 10

Наконец были измерены скорости переключения сопрограмм в управляемых средах HuaweiJDK, OpenJDK и языка Go. Как видно на слайде, HuaweiJDK выигрывает в скорости переключения у OpenJDK на всех измерениях, но проигрывает языку Go.

слайд 11

Так же была измерена скорость переключения потоков. Как видно из слайда, сопрограммы имеют лучшую скорость переключения, чем потоки на всем диапазоне измерений.

слайд 12

По сравнению с другими реализациями, сопрограммы в HuaweiJDK имеют неплохой результат по потреблению памяти: они обходят OpenJDK, но проигрывают сопрограммам в Go.

cлайд 13

Измерение потребления физической памяти потоков показало :), что сопрограммы имеют меньшее потребление памяти, чем потоки при любом их числе.

слайд 14

После проведения замеров можно сформулировать ключевые отличия потоков от сопрограмм. <пересказ>

<остальные слайды пересказываются>