слайд 1.

Типичная система обработки и сбора данных представляет собой 2 больших модуля. Один из них отвечает за сбор данных в режиме жесткого реального времени. Он состоит из набора устройств для измерения показаний, которые способны аккумулировать свои данные в локальную память. Затем эта информация передается по сети второму модулю. Он отвечает за пред обработку полученного набора данных и запись результатов на диск, либо в базу данных.

слайд 2

Второй модуль можно реализовать как систему мягкого реального времени. Для этих целей правильно использовать язык программирования Java, поскольку он упрощает разработку, так как обладает автоматической системой управления памятью, контролем ошибок при исполнении и так далее.

слайд 3

Работу с множеством каналов связи можно организовать с помощью их последовательного опроса. Но к сожалению такой способ труднее реализовывается по сравнению с выделением потока на обработку данных каждого канала. Но потоки имеют высокое потребление памяти и процессорного времени на переключение контекста, что ограничивает пропускную способность системы. Для такой задачи существует возможная альтернатива потокам - сопрограммы.

слайд 4

<пересказ>

слайд 5

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 6

<пересказ>

слайд 7

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

слайд 8

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 9

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 10

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

слайд 11

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 12

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

слайд 13

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

cлайд 14

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

слайд 15

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

<остальные слайды пересказываются>